Thomas J. Cova Ha
Kate Beard Andre
Michael F. Goodchi

LNCS 5266

Geograp
Informat

5th International Conferen
Park City, UT, USA, Septem
Proceedings

Thomas J. Cova   Harvey J. Miller
Kate Beard   Andrew U. Frank
Michael F. Goodchild (Eds.)

Geographic Information Science

5th International Conference, GIScience 2008
Park City, UT, USA, September 23-26, 2008
Proceedings

Springer

Volume Editors

Thomas J. Cova
Harvey J. Miller
University of Utah
Salt Lake City, UT 84112-9155, USA
E-mail: {cova, harvey.miller}@geog.utah.edu

Kate Beard
University of Maine
Orono, ME 944469-5711, USA
E-mail: beard@spatial.maine.edu

Andrew U. Frank
Vienna University of Technology
1040, Vienna, Austria
E-mail: frank@geoinfo.tuwien.ac.at

Michael F. Goodchild
University of California
Santa Barbara, CA 93106-4060, USA
E-mail: good@geog.ucsb.edu

Library of Congress Control Number: 2008934859

CR Subject Classification (1998): J.2, H.2.8, I.3.5, I.5.1, F.2.2, J.7

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

ISSN 0302-9743
ISBN-10 3-540-87472-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-87472-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12525546 06/3180 5 4 3 2 1 0

Preface

The GIScience conference series was founded in 2000 with the goal of providing a forum for researchers interested in advancing the fundamental aspects of the production, dissemination, and use of geographic information. The conference is held biannually and attracts people from academia, industry, and government across a host of disciplines including cognitive science, computer science, engineering, geography, information science, mathematics, philosophy, psychology, social science, and statistics. Following a very successful conference in Münster, Germany in 2006, this year's conference was held in Park City, Utah, USA, the prior site of the 2002 Winter Olympics and home to the annual Sundance Film Festival.

There are two forms of submission to the conference: full papers of 6000 words or less and extended abstracts of 500-1000 words for either a presentation or poster. This format was originally designed to capture the cultural difference between researchers who prefer to publish a peer-reviewed conference paper and those who would rather submit an abstract covering work in progress. This year 77 full papers were submitted and reviewed by 3 Program Committee members, of which 24 were selected for presentation and inclusion in this volume. Of the 115 extended abstracts that were submitted and reviewed by 2 Program Committee members, 47 were accepted for an oral presentation and 25 were accepted for presentation as a poster. The abstracts were published in a second booklet and are available on the GIScience website (http://www.giscience.org).

The breadth of new topics represented in this volume highlight the dynamic nature of GIScience. While traditional topics such as spatial relations, geographic dynamics, and spatial data types are still actively being advanced, new topics including geosensors, mobile computing, and Web mapping have come to the fore. The topics of navigation, networks, and location-based services also continued to be well represented in this year's submissions. While spatial information query and retrieval continues to be a hot topic, work advancing geo-ontologies has settled into the realm of a more conventional topic following its peak in 2004. Finally, in addition to the many sessions dedicated to presenting papers and works-in-progress, there were five keynote talks, eight pre-conference workshops, and a poster and wine session.

We would like to thank the many people that made GIScience 2008 possible. Thanks to the Program Committee for their enormous effort in reviewing the submissions. Oscar Larson and Doug Richardson from the Association of American Geographers helped tremendously with all aspects of the event from securing the venue to managing the on-line registration. Thanks also to the program sponsors who helped support both student and plenary speakers through travel grants. Finally, a special thanks to Melissa Warner in the Digitally Integrated Geographic Information Technologies (DIGIT) Lab, Department of Geography, University of

Utah for her webpage design and management and Laura Siebeneck who handled much of the outreach.

July 2008

Tom Cova
Harvey Miller
Kate Beard
Andrew Frank
Mike Goodchild

Organization

Program Chair

Thomas J. Cova	University of Utah

Program Co-chairs

Kate Beard	University of Maine
Michael F. Goodchild	University of California Santa Barbara
Andrew U. Frank	Vienna University of Technology

General Chair

Harvey J. Miller	University of Utah

Program Committee

Dave Abel	CSIRO
Pragya Agarwal	University of Leeds
Luc Anselin	Arizona State University
Walid Aref	Purdue University
Marc Armstrong	University of Iowa
Scott Bell	Brown University
Michela Bertolotto	University College Dublin
Ling Bian	University at Buffalo
Thomas Bittner	University at Buffalo
Dan Brown	University of Michigan
Gilberto Câmara	INPE
Nicholas Chrisman	Laval University
Christophe Claramunt	Naval Academy Research Institute
Helen Couclelis	University of California Santa Barbara
Isabel Cruz	University of Illinois at Chicago
Leila de Floriani	University of Genova
Matt Duckham	University of Melbourne
Max Egenhofer	University of Maine
Sara Fabrikant	University of Zurich
Stewart Fotheringham	National University of Ireland, Maynooth
Christian Freksa	University of Bremen
Mark Gahegan	University of Auckland
Ralf Güting	University of Hagen

Francis Harvey	University of Minnesota
Gerard Heuvelink	Wageningen University
Stephen Hirtle	University of Pittsburgh
Piotr Jankowski	San Diego State University
Chris Jones	Cardiff University
Marinos Kavouras	National Technical University of Athens
Menno-Jan Kraak	ITC
Werner Kuhn	University of Münster
Mei-Po Kwan	The Ohio State University
Paul Longley	University College London
Daniel Montello	University of California, Santa Barbara
Atsuyuki Okabe	University of Tokyo
Harlan Onsrud	University of Maine
David O'Sullivan	University of Auckland
Dimitris Papadias	University of Science and Technology, Hong Kong
Jonathan Raper	City University of London
Martin Raubal	University of California, Santa Barbara
Andrea Rodríguez	Universidad de Concepción
Nadine Schuurman	Simon Fraser University
Wen-zhong (John) Shi	Hong Kong Polytechnic University
Takeshi Shirabe	Vienna University of Technology
Ashton Shortridge	Michigan State University
Kathleen Stewart-Hornsby	University of Iowa
Paul Sutton	University of Denver
Ming Tsou	San Diego State University
Marc van Kreveld	Utrecht University
Rob Weibel	University of Zurich
Stephan Winter	University of Melbourne
Mike Worboys	University of Maine
May Yuan	University of Oklahoma

Additional Reviewers

Jonathan Arundel
Thomas Behr
Scott Bridwell
M. A. Cameron
Elena Camossi
H.G. Elmongui
M.Y. Eltabakh
Herman Haverkort
Yuxia Huang
Margaret Kalackska
Carsten Kessler

Joe Messina
Edzer Pebesma
Kai-Florian Richter
Y.N. Silva
Yunhui Wu
Olga Yermakhanova
Tong Zhang

Sponsoring Institutions

Department of Geography, University of Utah
Association of American Geographers (AAG)
National Center for Geographic Information and Analysis (NCGIA)
University Consortium for Geographic Information Science (UCGIS)
Environmental Systems Research Institute (ESRI)
Oak Ridge National Labs (ORNL)
Taylor & Francis

Table of Contents

Query Responsive Index Structures

Ludger Becker[1], Hannes Partzsch[1], and Jan Vahrenhold[2]

[1] Fachbereich Mathematik und Informatik, Institut für Informatik, Westfälische Wilhelms-Universität Münster, Einsteinstr. 62, 48149 Münster, Deutschland
[2] Fakultät für Informatik, Informatik XI, Technische Universität Dortmund, Otto-Hahn-Str. 14, 44227 Dortmund, Deutschland

Abstract. In this paper, we generalize the notion of self-adapting one-dimensional index structures to a wide class of spatial index structures. The resulting *query responsive index structures* can adapt their structure to the users' query pattern and thus have the potential to improve the response time in practice. We outline two general approaches to providing query responsiveness and present the results in terms of the well-known R*-tree. Our experiments show that depending on the query pattern significant improvements can be obtained in practice.

1 Introduction

Accessing spatial databases, whether directly or indirectly, is no longer an operation available to expert users only but has become an integral component of many non-expert users' routine. Prominent scenarios are using online navigation systems, digital satellite imagery, or online encyclopedias that contain geo-referenced data.

Depending on the type of database, the users' queries may exhibit a strongly local behavior that changes over time. For example, a geographic information system linked to a news feed will receive most queries about those geographic regions that appear in the headlines, and a traffic information system is likely to receive a large number of queries about regions where traffic jams occur. Also, other types of databases that use (high-dimensional) index structures may exhibit this behavior, e.g. biological or multimedia databases. It is thus desirable that spatial indexes adapt their structures to provide faster response times for areas of current interest.

As spatial databases continue to increase both coverage and resolution, many spatial index structures have reached a size so large that the only feasible update mechanism is through bulk operations—see e.g. [6,12,19,31]. Thus, such spatial index structures *de facto* can been considered static in between these updates, and this rules out all restructuring approaches that reorganize the index structure when triggered by an update operation.

In this paper, we introduce the notion of *query responsive index structures* that adapt their structure to the users' query pattern while answering queries. To the best of our knowledge, this is the first approach to self-organizing spatial index structures that is formulated and evaluated in a database setting.

The remainder of this paper is structured as follows: After a review of the related work (Section 2), Section 3 presents two different approaches to providing query responsiveness. In Section 3.1, we show how to integrate statistical information into a spatial index structure, and in Section 3.2, we employ a secondary data structure that can be attached to and detached from an index structure as needed. The experimental evaluation of these structures and a comparison with a benchmark structure is summarized in Section 4.

2 Self-organizing Index Structures

In this section, we give a brief account of related work on the design and analysis of self-organizing index structures.

2.1 One-Dimensional Index Structures

Most one-dimensional index structures heavily build on the fact that a total order on the data objects is known; an important exception are hash-based data structures which are beyond the scope of this paper. Knuth [21] gave a quadratic algorithm based on dynamic programming that, for a given set of objects and a given query distribution, constructs an optimum binary search tree that minimizes the overall query cost. The quality of faster heuristics was later analyzed by Mehlhorn [24] and Bitner [11].

The investigation of the performance of self-organizing unordered lists and (binary) search trees has received a considerable amount of attention, both from a theoretical and a practical point of view—see, e.g, [3,9,11]. Perhaps the best-known self-organizing binary search tree is the *splay tree* proposed by Sleator and Tarjan [28]. Also, randomization has been shown to improve the performance of splay trees both in practice and in expectation [2,14].

2.2 Higher-Dimensional Index Structures

The classic higher-dimensional index structure is the *R-tree* [17,23,30] which will be described in more detail below. This structure is primarily used for data in low-dimensional Euclidean spaces and served as a “role model” for a variety of structures for data objects in high-dimensional and/or metric spaces—see the recent book by Samet [26] for a comprehensive survey.

Almost all tree-based index structures belong to the class of so-called *grow-and-post-trees* [22], that is they allow for an insertion at leaf level and grow and shrink at the top of the tree. Many spatial index structures, including the R-tree and its variants, also belong to a subclass of grow-and-post-trees referred to as *overlapping-predicate-trees* [31]. In such trees, it is also permissible to insert subtrees whose height is lower than the height of the original tree. This, however, requires that all elements stored in any subtree fulfill the same predicate (e.g. they are all contained in a certain area of the data space) and that the predicates

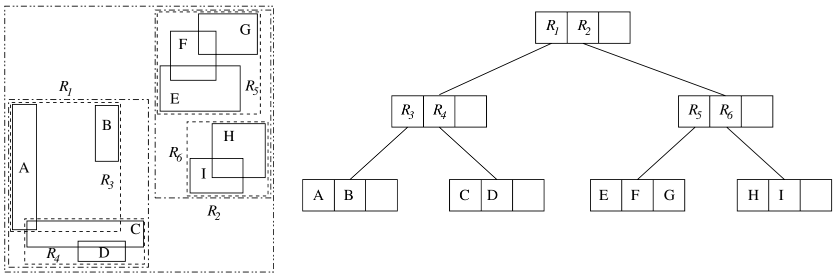


Fig. 1. R-tree for data rectangles A, B, C, ..., I [6]. The tree in this example has maximum fanout $B = 3$.

valid for two different subtrees are allowed to be non-disjoint.[1] While the techniques described in this paper can be applied to any overlapping-predicate-tree, we simplify the exposition by restricting ourselves to two-dimensional R-trees and their variants.

Description of the R-Tree. The R-tree, originally proposed by Guttman [17], is a height-balanced multiway tree similar to a B-tree. An R-tree stores d-dimensional data objects approximated by their axis-parallel minimum bounding rectangles. For ease of presentation, we restrict the exposition to the situation $d = 2$ and assume that each data object itself is an axis-parallel rectangle.

The leaf nodes in an R-tree contain $\Theta(B)$ data rectangles, where B is the maximum fanout of the tree. Internal nodes contain $\Theta(B)$ entries of the form (Ptr,R), where Ptr is a pointer to a child node and R the minimum bounding rectangle covering all rectangles which are stored in the subtree rooted in that child. Each entry in a leaf stores a data object or, in the general setting, the bounding rectangle of a data object and a pointer to the data object itself. Fig. 1 shows an example of an R-tree for a set of two-dimensional rectangles.

Updates and Queries on R-Trees. To insert a new rectangle r into an already existing R-tree with root v, we select the subtree rooted at v whose bounding rectangle needs least enlargement to include the new rectangle. The insertion process continues recursively until a leaf is reached, adjusting routing rectangles as necessary. If a leaf overflows due to an insertion, a rebalancing process similar to B-tree rebalancing is triggered, and therefore R-trees also grow and shrink only at the top. The insertion path depends not only on the heuristic chosen for breaking ties in case of non-unique subtrees for recursion, but also on the objects already present in the R-tree. Hence, there is no unique R-tree for a given set of rectangles, and different orders of insertion for the same set of rectangles usually result in different R-trees.

[1] An important class of grow-and-post-trees that does not belong to the subclass of overlapping-predicate-trees is the class of B-trees; here the predicate is an interval on the real line, and no two subtrees' intervals are allowed to overlap.

During the insertion process, a new rectangle r might overlap the routing rectangles of several subtrees of the node v currently visited. However, the rectangle r is routed to exactly one such subtree. Since the routing rectangle of this subtree is extended to include r, the routing rectangles stored within v can overlap. This overlap directly affects the performance of R-tree query operations: When querying an R-tree on N data objects to find all rectangles overlapping a given query rectangle r, we have to branch at each internal node into all subtrees whose minimum bounding rectangle overlaps r. In the worst case, the search process has to branch at each internal node into all subtrees which results in $\mathcal{O}(N/B)$ nodes being touched—even though the number of reported overlapping data rectangles might be much smaller. Intuitively, it is thus desirable that the routing rectangles stored within a node overlap as little as possible.

As mentioned above the insertion of a new rectangle can increase the overlap between routing rectangles stored in the same internal node and thus can also increase search time. To alleviate this effect, several heuristics have been proposed for choosing the leaf into which a new rectangle is inserted, and for splitting nodes during rebalancing [8,16,18,27]. Among these heuristics, the *R^*-tree* [8] and the *Hilbert R-tree* [18] have emerged as especially successful variants, and the R^*-tree is the most widely implemented data structure that also serves as a canonical benchmark.

Related work. The R-tree and its variants, most notably the R^*-tree, attempt to adjust their internal structure to dynamically changing data objects during update operations. Whenever a node is split or merged as a result of an insert or delete operation, the "best" way of doing this is determined according to one of several heuristics [16]. One of the practically most efficient ways of updating the internal structure after a delete operation is the *forced reinsertion* used in the R^*-tree [8].

Adaptive R-trees (and *B-trees*) [29] aim at reducing the overall query time by optimizing the layout and node size of the tree. Following the example of the X-tree [10], they allow for "large" node sizes to reduce expensive seek operations. Their approach is to maintain query histograms and to use this information to compute an optimal (according to the predicted access pattern) page size for nodes created during split or merge operations. Note that this approach and all of the above approaches require split or merge operations to trigger the restructuring process, i.e. they only work for dynamically changing data sets.

The *Cost-based Unbalanced R-tree* [25] assumes a given (global) query distribution that is known to the data structure and tries to construct a tree that is locally optimal with respect to this distribution. As a consequence, the resulting tree may have both over- and underfull nodes and data objects stored at any level of the tree. Unfortunately, no non-trivial bounds on the fanout and height of the tree are given, and thus the authors recommend to use this structure in an in-memory environment. Also, it remains to be investigated to which extent of practical efficiency this structure can be used for answering other types of queries, e.g. spatial join queries.

To the best of our knowledge, the only spatial data structure that is able to restructure itself according to an observed query pattern is the *Spatial Splay Tree* [13]. This data structure essentially implements a linear quadtree using a binary splay-tree. Since the maximal fanout of such a tree is two, the efficient use of this data structure is restricted to in-memory environments.

3 Query-Responsive Index Structures

In this section, we present two approaches to setting up a spatial index structure such that it is able to adapt to a specific access pattern. The proposed modifications do not change the interface or the internal structure and thus do not affect the use of the structure, e.g. for query or join operations. While we use an R*-tree to illustrate the concepts we point out that they can be applied to any overlapping-predicate-tree.

3.1 Remembering the Past: The Matrix-R-Tree

The first of our approaches is based on the following difference between range queries in B-trees and in higher-dimensional overlapping-predicate-trees (e.g. R*-trees): It is well-known that a range query in a B-tree cannot overlap more than two leaves of the tree without reporting any data object. For the case of a d-dimensional R*-tree, or more generally, for any spatial index structure that does not replicate data objects, a range query may overlap $\Theta((N/B)^{1-\frac{1}{d}})$ leaves without reporting any data object [20] (here, N is the number of data objects and B is the maximal fanout of the tree).

The worst-case scenarios for R*-trees can easily be seen to exhibit one common feature: many bounding boxes overlap and thus there are many branchings during the execution of a query. In general, such overlaps cannot be avoided completely simply because of the position of the data objects in space—see e.g. [1,5]. From a practical point of view, however, we would like as few branchings as possible to occur for regions that are frequently visited by queries.

Definition of the Matrix-R-Tree. Since for most practical applications the queries (or their distribution) are not known in advance and/or may vary over time, we need to establish other means of even detecting that frequent branchings occur. We therefore propose to augment each node ν of an R*-tree such that it can record the number of queries that branched at ν. To allow for a meaningful processing of these statistics, we also need to record to which nodes a query branched.

Definition 1. *A* Matrix-R-Tree *is an R*-tree $\mathcal{T}$ that, for each non-leaf node $\nu \in \mathcal{T}$, maintains information about how many queries branched to more than one child of ν: for each pair (a_i, a_j) of children of ν, we record how many queries branched to both a_i and a_j. For each pair (a_i, a_j) of data objects stored in a leaf node, we record how many queries reported both a_i and a_j.*

The above definition allows us to store the counters in an upper triangular matrix storing only the counters for (a_i, a_j) with $i < j$. Ideally, we would like to maintain the above statistical information not only for pairs but for *each* potential subset of children of ν. This, however, would result in a prohibitively excessive space requirement of 2^B counters where B is the maximal allowed fanout for $\mathcal{T}$: for $B = 50$ and one byte per counter, we would need to maintain $2^{50} = 1024 \cdot 1024$ GB of statistical information *per node*. Instead, to record multi-way branchings of queries, we decompose each branching to k children $(2 \leq k \leq B)$ into $\binom{k}{2}$ pairs of children branched to and record these pairs.

If we can maintain the statistic for node ν in the same block as (the routing information of) ν, we do not increase the number of disk accesses for answering a query. This, however, comes at the cost of a restricted space allowed for the statistic. Assuming a page size of P bytes, four eight-byte coordinates and one eight-byte identification per data rectangle, a two-byte counter per pair of children, and twelve byte of other statistical information per node, the maximal fanout B is given as follows:

$$B \;=\; \max_{n \in \mathbb{N}} \left\{ 40 \cdot n + 2 \cdot \frac{(n-1)n}{2} + 12 \;\leq\; P \right\}$$

Solving this for several values of P relevant in practice results in the following table that, for a given P, compares the maximal fanout B_M of a Matrix-R-Tree with the maximal fanout B of a standard R*-tree.

Page Size P	2 KB	4 KB	8 KB	16 KB	32 KB	64 KB
Fanout B_M	29	47	73	109	162	237
Fanout B	50	102	204	409	818	1638

Previous studies on the practical performance of R*-tree, e.g. [5,6,8,17,15], have noted that the most common values used for R-trees in practice are $B = 50$ and $B = 100$. This choice is mainly due to a trade-off between internal-memory computation time (especially for node-splits and spatial join operations) and height of the tree. The above table shows that, with current page sizes between 8 KB and 64 KB, we can easily store both the routing elements as well as the statistics in one page, i.e. we can access both the routing elements and the statistics using at most one page access.

Operations on a Matrix-R-Tree. All construction and update operations in a Matrix-R-Tree are basically the same as in an R*-tree with the obvious modifications needed for maintaining the correctness of the statistics: Whenever an entry a_i is added to (deleted from) a non-leaf node ν, we have to add (delete) the corresponding row and column to (from) the matrix storing the branching counters. Since we can choose the maximal fanout of the tree such that all routing information and the matrix fit into one page, these modifications do not cause any extra page accesses.

As stated above, the query operation keeps track of how many (and which) nodes it branched to during its execution. In the base version of our data structure, we would increment the counter (a_i, a_j) by one whenever the query branches to both the children a_i and a_j of the current node. In a modified version, called IncrementIntersection, we also take into account how much the query region R overlaps the region R_ν associated with the node ν and increment the counter(s) not by one but by the percentage of the area of R_ν overlapped by R, that is by $s_{R_\nu}(R) := \text{area}(R_\nu \cap R)/\text{area}(R_\nu)$.

Reorganizing a Matrix-R-Tree. If, during a query operation, a counter (a_i, a_j) *overflows*, i.e. becomes at least as large as a given threshold μ, we trigger a reorganization of the tree. We present two different types of reorganization approaches: *split-based* approaches and *reinsertion-based* approaches.

Split-Based Reorganization. The most natural way to react to a overflowing counter (a_i, a_j) of a node ν would be to collect the entries stored in the nodes a_i and a_j and to try to find a better distribution of these entries into two nodes a'_i and a'_j. Such an approach would be advisable in a dynamic environment where the original distribution found during the split that created a_i and a_j is no longer valid due to interleaving insert and delete operations. One option for the newly created nodes a'_i and a'_j is to replace the original children a_i and a_j in their parent ν (we call this variant MSplitNodes-Parent), the other option would be to use the reinsert operation of the R*-tree to reinsert a'_i and a'_j into the tree at the appropriate level making sure to handle a potential underflow in ν (we call this variant MSplitNodes-Insert). Both variants can be combined with the IncrementIntersection approach described above.

While the above approach is relatively easy to describe and implement, its main drawback seems to be that we can "reset" only one counter per reorganization. The variant MSplitMultiplePairs tries to alleviate this effect: it uses a greedy algorithm to find a collection of independent overflowing counters, i.e. a collection of overflowing counters (a_i, a_j) such that each child of the current node ν contributes to at most one such counter. For each of these counters, we then run the above MSplitNodes-Parent algorithm (using the MSplitNodes-Insert algorithm is likely to result in an underflow in μ's parent).

If we replace the check for overflowing counters by a (biased) coin flip, we obtain what we call the RandomSplit variant.[2] In this variant, the query method of the R*-tree is augmented to also return a set $\mathcal{A}$ of nodes that should be used for reorganizing. As long as no such set has been determined, the query method flips a (biased) coin at each node visited. If the coin flip indicates so, the set $\mathcal{A}$ is constructed by selecting all children of the current node ν whose bounding boxes overlap the query rectangle R. After the query algorithm terminates, the set $\mathcal{A}$ is either empty or contains a subset of the children of exactly one internal node, and this set is then used for reorganization as described above. For the success

[2] This variant uses the reorganization subroutines of the Matrix-R-Tree but is not a Matrix-R-Tree in the strict sense since it does not need to maintain counters at all.

Table 1. Name and short description of each of the split-based approaches

Name	Description
MSplitNodes-Parent	Split into two nodes, reinsert at parent.
MSplitNodes-Insert	Split into two nodes, reinsert at root.
MSplitNodesII-Parent	Split into two nodes, reinsert at parent (counters are updated by IncrementIntersection).
MSplitNodesII-Insert	Split into two nodes, reinsert at root (counters are updated by IncrementIntersection).
MSplitMultiplePairs	Collect independent pairs of nodes, split each pair separately, reinsert at parent.
RandomSplit-Parent	Trigger MSplitNodes-Parent by (biased) coin flip.
RandomSplit-Insert	Trigger MSplitNodes-Insert by (biased) coin flip.
RandomSplit-Intersection	Trigger MSplitNodes-Parent by (biased) coin flip that takes into account $S_{R_\nu}(R)$.
RandomSplitMultiplePairs	Trigger MSplitMultiplePairs by (biased) coin flip.

probability p of the coin flip, we can either use a fixed probability $p \in [0,1]$ for each node or an adaptive probability $p := (p' + s_{R_\nu}(R))/2$ where $p' \in [0,1]$ is a (global) probability and $s_{R_\nu}(R)$ is the relative area of the bounding rectangle R_ν of ν overlapped by the query rectangle R.

All split-based approaches are summarized in Table 1.

Reinsertion-Based Reorganization. Since most of the practical efficiency of the R*-tree is due to the *forced reinsertion* triggered by overfull nodes, we now describe an approach to mimic this behavior for query operations.

The main subroutine used in all variants of this approach is the algorithm FindRectangles (Algorithm 1). This subroutine recursively finds all data

Algorithm 1. FindRectangles(ν): Returns a set $\mathcal{A} = \{a_1, \ldots, a_\ell\}$ of data objects stored in the subtree rooted at ν such that one of the counters associated with each a_i overflows.

```
FR1: If $\nu$ is not a leaf node then do the following:
       – For each child $a_i$ of $\nu$ do the following:
           • If $(a_i, a_j) \geq \mu$ for some child $a_j$ of $\nu$ then
             set $\mathcal{A} \leftarrow \mathcal{A} \cup$ FindRectangles($a_i$).
FR2: If $\nu$ is a leaf node then do the following:
       – For each data object $a_i$ of $\nu$ do the following:
           • [Report those data objects that contribute to overflowing counters.]
             If $(a_i, a_j) \geq \mu$ for some data object $a_j$ in $\nu$ then set $\mathcal{A} \leftarrow \mathcal{A} \cup \{a_i\}$.
FR3: Return the set $\mathcal{A}$.
```

Algorithm 2. ChooseGroup: For a given node ν with B_ν children, this algorithm returns a set $\mathcal{A} = \{a_i, a_{j_1}, \ldots, a_{j_k}\}$ of children of ν such that $(a_i, a_{j_\ell}) \geq \mu$ for $1 \leq \ell \leq k$. Furthermore, all corresponding counters are reset to zero.

```
CG1: [Initialize the algorithm.] Set $i \leftarrow 0$, $j \leftarrow 1$, and $\mathcal{A} \leftarrow \emptyset$.
CG2: [Find a row $i$ such that $a_i$ and some $a_j$ were branched to simultaneously at least
    $\mu$ times.]
    While ($i < B_\nu$ and $(a_i, a_j) < \mu$) do:
      – [Advance to next column.] Set $j \leftarrow j + 1$.
      – [Advance to next row.] If $j > B_\nu$ then set $i \leftarrow i + 1$ and $j \leftarrow i + 1$.
CG3: [Collect entries from row $i$ (if it exists).]
    If ($i < B_\nu$) then
      – [Add $a_i$ to result set.] Set $\mathcal{A} \leftarrow \mathcal{A} \cup \{a_i\}$.
      – [Add $a_j$ to result set and iterate.] While ($j < B_\nu$) do:
          If ($(a_i, a_j) \geq \mu$) then set $\mathcal{A} \leftarrow \mathcal{A} \cup \{a_j\}$ and $j \leftarrow j + 1$.
CG4: [Reset the counters of the elements collected.] Let $\mathcal{I}$ be the set of indices of the
    elements in $\mathcal{A}$. Set $(a_i, a_j) \leftarrow 0$ for $(i, j) \in \mathcal{I} \times \mathcal{I}, i < j$.
CG5: Return the set $\mathcal{A}$.
```

objects a_i (i.e. entries stored in leaves) of the current subtree that have at least one overflowing counter (a_i, a_j).[3]

In the basis version (MBasis) of the reinsertion-based algorithm a counter (a_i, a_j) is incremented by one if a query branches to both (a_i, a_j). Just as in the above split-based approaches, there is a variant (MIntersection) of the algorithm that takes into account how much the query rectangle R overlaps the bounding box R_ν of the current node ν and increments the counter accordingly, that is by $s_{R_\nu}(R) = \text{area}(R_\nu \cap R)/\text{area}(R_\nu)$.

In any case, an overflowing counter first triggers a call to ChooseGroup (Algorithm 2) to find a group $\mathcal{A} = \{a_i, a_{j_1}, \ldots, a_{j_k}\}$ of children that have a common overlap and overflowing counters (a_i, a_{j_ℓ}).

After all (recursive) queries initiated by a user running the standard query algorithm have been answered and the group of nodes contributing to overflowing counters is not empty, we process this group. For each child a_i in the resulting group, a call to FindRectangles(a_i) produces all data objects in the tree rooted at a_i that contribute to one overflowing counter. These data objects then are deleted from the tree and reinserted at the root.[4]

To possibly reduce the number of query operations needed to construct the set of data objects to be reinserted, we also investigate the following approach: We first invoke the algorithm ChooseGroup to find a collection $\mathcal{E}$ of bounding boxes. Based

[3] Recall that, since the relation "a query branches to a_i and a_j" is symmetric, we only need to store and access counters of the form (a_i, a_j) where $i < j$.

[4] Here, we see why it is important to have counters also in the leaves—otherwise we would not be able to select only those data objects from any given leaf that contribute to overflowing counters, i.e. have been touched by multiple queries that branched "too much"; instead, we would delete and re-insert too many data objects.

Table 2. Name and short description of each of the reinsertion-based approaches

Name	Description
MBasis	Reinsert all data objects that correspond to overflowing counters.
MIntersection	Reinsert all data objects that correspond to overflowing counters and take into account $s_{R_\nu}(R)$.
MBR(Group)	Reinsert all data objects in the minimal bounding rectangle of data objects that correspond to overflowing counters.
MBR(Intersection)	Reinsert all data objects in the minimal bounding rectangle of the pairwise intersection of all data objects that correspond to overflowing counters.
MBR(FindRectangles)	Reinsert all data objects in the minimal bounding rectangle of the result of FindRectangles.

upon this collection, we construct a single rectangle R. We then delete *all* data objects that overlap R from the index structure and reinsert them immediately afterwards. There are at least three possible variants of this approach: (1) the rectangle R is constructed as the minimum bounding rectangle of $\mathcal{E}$ (MBR(Group)), (2) the rectangle R is constructed as the minimum bounding rectangle of all pairwise intersections of rectangles in $\mathcal{E}$ (MBR(Intersection)), and (3) the rectangle R is constructed as the minimal bounding rectangle of all *data objects* that are stored in the (subtrees rooted in the) nodes in $\mathcal{E}$ and that contribute to overfull counters (MBR(FindRectangles)).

All reinsertion-based approaches are summarized in Table 2.

3.2 Recalling the Past: The Query R-Tree

The second of our approaches to providing query responsiveness is based upon a recording strategy: We maintain a second index structure that keeps track of which queries have been answered so far—and how often this was the case. For our running example, the R*-tree, it is particularly easy to see that we can actually use another R*-tree to record these queries. For the general case of an overlapping-predicate-tree, we need to maintain an index structure that can store the type of objects that define a query, and in many situations this will be an instance of the same class of index structures.

Definition 2. *A* Query-R-Tree *is an R*-tree $\mathcal{T}$ (*data tree*) that is bundled with a second index structure (*query tree*) that records the queries on $\mathcal{T}$ that have been answered.*

All update operations on the data tree are performed exactly as in an R*-tree.

Reorganizing a Query-R-Tree. In the basis version of a Query-R-Tree, we answer the query on the data tree and record it by inserting the query rectangle into the query tree; this data structure stores the queries (rectangles) augmented by a counter. Initially, each counter is set to one, and whenever we find a rectangle that has been inserted before, we do not insert another copy but instead increment the counter by one. As soon as a counter overflows, i.e. becomes larger than a given threshold μ, we trigger a reorganization of the data tree.

If a counter overflow is detected for a rectangle R, we query the data tree using R and obtain a set $\mathcal{N}$ of rectangles. We then *bulk delete* $\mathcal{N}$ from the data tree and *bulk (re-)insert* them using the algorithm of Kamel *et al.* [19] that uses the Hilbert space-filling curve to obtain a good packing of the data rectangles into leaves.[5] Finally, we reset the counter of R in the query tree.

This first approach is complemented by an approach which takes into account that a query rectangle may intersect other query rectangles. This approach, which is called QIntersection, handles a query as follows. After the query has been answered, we perform a second query using the same query rectangle R: this time, we query the *query tree* and report the set $\mathcal{R}$ of all "old" query rectangles that overlap R. For each rectangle $R' \in \mathcal{R}$, we increment its counter by $s_{R'}(R) := \text{area}(R' \cap R)/\text{area}(R')$. For each rectangle whose counter overflows, we trigger a reorganization of the data tree as described above. In a variant of this approach, we do not reorganize with each such rectangle but with the minimal bounding rectangle of all such rectangles—this variant is called QIntersectionMBR.

Finally, we also describe a randomized version of the query-based reorganization approach (called QRandom). Similar to the randomized version of the Matrix-R-Tree, this approach does not store statistical information at all (thus, there is only one index structure, namely the data tree). Instead, this method simply starts a reorganization using the current query rectangle (as described above) based upon a (biased) coin flip.

All query-based approaches are summarized in Table 3.

Comparison of Both Approaches From a Database Perspective. The main difference between the two proposed methods is that the Matrix-R-Tree augments the internal nodes of the index structure whereas the Query-R-Tree maintains a secondary index structure for the statistics. As a consequence, the "Query" approach considers the primary index structure as a black box and thus provides a "cleaner" separation between statistics and data. Furthermore, the secondary index structure can be attached to resp. detached from the primary structure at any time without affecting the integrity of the former structure, i.e. without the need to obtain a lock.

The "Matrix" approach, on the other hand, can maintain the statistics *on-the-fly*: statistical information is updated only in nodes that are currently accessed by the algorithm. This means there is no additional overhead in terms of node accesses, and the complexity of obtaining locks during the query process is exactly the same as it would be for the primary structure without statistical information.

[5] While in general this bulk insertion can lead to highly overlapping leaf nodes [6], it is well suited for our situation: the data tree to be inserted into cannot contain *any* rectangle since all these rectangles are in the set $\mathcal{N}$.

Table 3. Name and short description of each of the query-based approaches

Name	Description
QBasis	Increment the counter for an exact match of the query rectangle. Delete and bulk (re-)insert all data objects inside any rectangle R' whose counter overflows.
QIntersection	Increment the counter for each query rectangle R' that overlaps the query rectangle by $s_{R'}(R)$. Trigger QBasis for all rectangles whose counters overflow.
QIntersectionMBR	Increment the counter for each query rectangle R' that overlaps the query rectangle by $s_{R'}(R)$. Trigger QBasis for the minimal bounding rectangle of all rectangles whose counters overflow.
QRandom	Trigger QBasis by (biased) coin flip.

Furthermore, no extra main memory, e.g. for the secondary structure as a whole or pinned parts of it, is needed.

The above differences seem to be merely a matter of personal preference and some of them are relevant only in concurrent environments. What is of much more practical importance is the actual query performance of both structures, and this is discussed in the next section.

4 Experimental Evaluation

In this section, we report on an experimental study undertaken to evaluate the practical efficiency of the proposed methods. We note that, to the best of our knowledge, this is the first experimental study on self-adjusting spatial index structures (Cobb *et al.* [13] do not report any experimental results).

Setup. The implementation of all data structures was done in `C++` using the `TPIE` library [4] for interfacing accesses to external memory. The base implementation for the R*-tree was also taken from `TPIE`, in particular, we used the implementation that served as the reference implementation in previous studies [5,6,7]. The measure of performance was the number of page accesses, and we did not employ any pinning or buffering strategies.

Query Types. In our experiments, we investigated three classes of queries representative of three application scenarios where users' queries may exhibit a strongly local behavior that changes over time.

Identical queries: In this scenario, a small set of disjoint queries is asked multiple times. This scenario models a situation where many users use a predefined set of query ranges. An example of such a situation is a geographic information system linked to a news feed, where selecting a text-based representation of a geographic region translates to issuing a pre-defined query operation on the database.

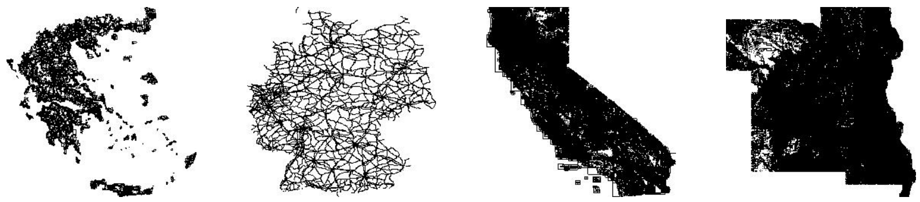


Fig. 2. Some of the data sets used for the experimental evaluation

- **Pairs of queries:** In this scenario, a small set of pairs of queries is asked multiple times; the overlap between the elements of each pair is between 13% and 75%, in one pair, one query is contained in the other. This extends the above situation by zoom-and-pan operations, e.g. by refining a query region. Also, this type of queries is well-suited to investigate the effect of possibly counteracting reorganizations.
- **Similar queries:** In this scenario, many highly overlapping non-identical queries are asked. This scenario models a geographic information system that allows for web-based access and thus interacts with users who ask queries by drawing query rectangles using a graphical front-end to the database.

We deliberately decided not to use randomly generated queries; such queries do not exhibit any coherence representative of the application scenarios that could benefit from query responsive data structures.

Data Sets. The data sets for our experimental evaluation were taken from the "RTreePortal" [30]. We used data sets from the complete range available, i.e. data sets having between 23,000 and 560,000 data objects. Some of data sets representative of different sizes and distributions are shown in Fig. 2.

For the experiments discussed in the next sections, we averaged the query performance across all data sets. To obtain index structures of non-trivial height, we chose the maximal fanout of the structures as $B = 10$ (in Section 4.3, we also report on experiments with higher fanouts).

4.1 Evaluation of the Matrix-R-Tree

In this section, we report on the performance of the Matrix-R-Trees relative to the R*-tree. For "identical queries", we repeated the same query 1000 times, for "pairs of queries", we alternatingly repeated the same two queries 500 times each, and for "similar queries" we used 200 manually entered queries. The threshold was set to $\mu := 5$, and for the randomized versions, we used $p := 1/\mu = 0.2$ resp. $p' := 1/4 = 0.25$ (for RandomSplit-Intersection).

Split-Based Matrix-R-Trees. The experiments for split-based Matrix-R-Trees, summarized in Fig. 3, consistently show an improvement of 3–12% over

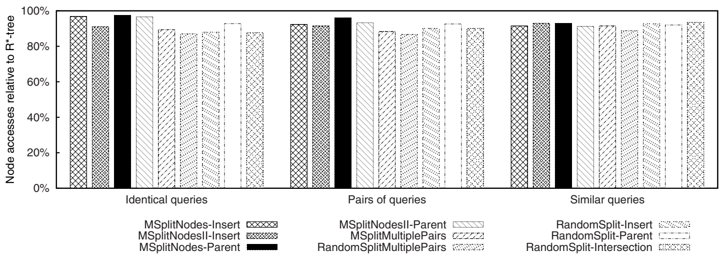


Fig. 3. Performance of split-based Matrix-R-Trees

the R*-tree across all tested variants and types of queries. Not surprisingly, the IncrementIntersection-variant outperform (on average) the variants that do not take into account the similarity measure $s_{R_\nu}(R)$. The best performance across all data sets with an average improvement of 10–12% is obtained by the two variants MSplitMultiplePairs and RandomSplitMultiplePairs that try to perform multiple splits at a node where a counter overflow is detected. A large percentage of the reorganizations triggered in the other variants did not lead to any structural change (see also Section 4.3).

We also investigated the performance for mixed query sets, i.e. set of queries that contain "identical queries", "pairs of queries", and "similar queries". In this experiment, we constructed a query set of 200 queries that was repeated 20 times. The data structures used in this experiment were representative of the structures that do not perform multiple splits at once (such that we could actually see the effects of a single reorganization). Even though the different types of queries seem to counteract each other, Fig. 4 shows that the reorganization continuously results in improved query performance. This also indicates that performing multiple splits is indeed beneficial for the query performance—see also Fig. 3.

Reinsertion-Based Matrix-R-Trees. The results for reinsertion-based Matrix-R-Trees (see Fig. 5) clearly demonstrate the potential of query-responsive index structures: For the "best-case" scenario of identical queries, the basis

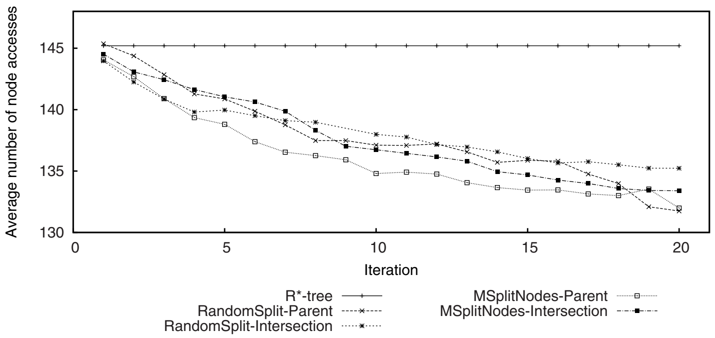


Fig. 4. Performance for mixed query sets

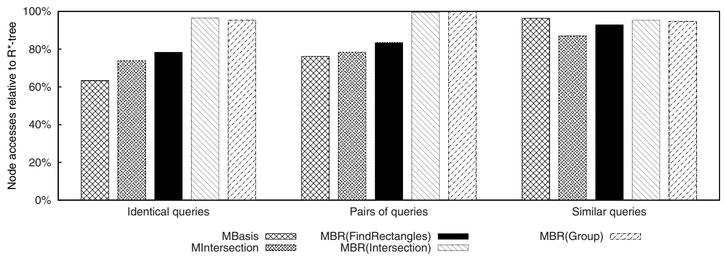


Fig. 5. Performance of reinsertion-based Matrix-R-Trees

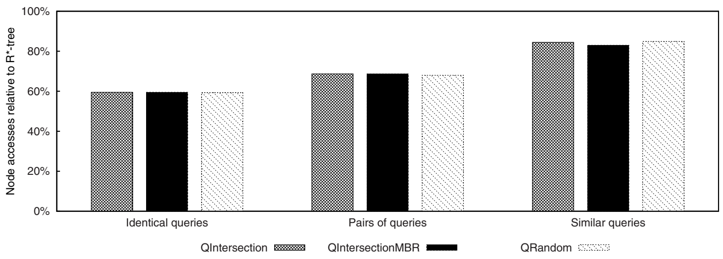


Fig. 6. Performance of Query-R-Trees

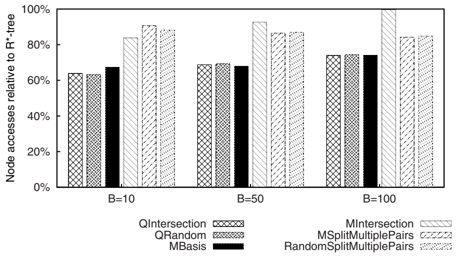
version obtains a speed-up of roughly 35%, and over all scenarios, the IncrementIntersection-variant leads to an improvement of 12–25%. On the other hand, the variants that try to reduce the work by using minimal bounding rectangles resulted in inferior query performance and were excluded from further studies.

4.2 Evaluation of the Query-R-Tree

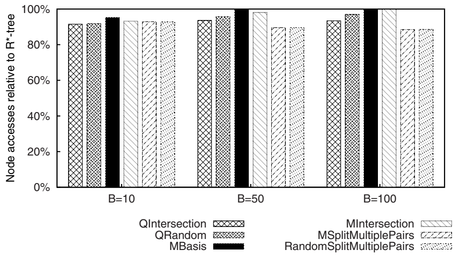
The results for Query-R-Trees show an even better performance for this type of query responsiveness. Depending on the type of query, these structures' query performance (including the cost for accessing the query tree) is 20–40% better than the performance of an R*-tree—see Fig. 6.

4.3 Influence of Fanout and Query Area

Fanout. In the initial experiments, we used a maximal fanout of $B = 10$ to obtain trees with non-trivial height and a threshold of $\mu = 5$ which, as discussed above, led to many effectless reorganizations. Thus, in a repetition of these experiments we used fanouts relevant in practice, i.e. $B = 50$ and $B = 100$, and increased the threshold to $\mu = 20$ for all other structures except for MSplitMultiplePairs and RandomSplitMultiplePairs (these structure should perform as many reorganizations as possible). The results of this set of experiments (Fig. 7) showed that there is a very moderate decrease in query performance



(a) Identical queries.



(b) Similar queries.

Fig. 7. Performance of Matrix-R-Trees and Query-R-Trees

for higher fanouts (i.e. shallower trees). Overall, the relative performance of the different variants stays the same, and they still outperform the R*-tree.

Query Area. To investigate a possible correlation of the query performance with the area covered by the query rectangle (relative to the area covered by the root of the tree), we ran a small set of experiments on the “TIGER/Streams” data set [30]. The query rectangle was centered in the data set and its relative size varied between 0.5% and 45%. Fig. 8 shows that the query performance does not seem to correlate with the size of the query unless the query area is so small that only a very small number of nodes is affected at all. If the relative size of the query area, however, is larger than 2%, the Query-R-Tree performs significantly better than both its opponents.

4.4 Reorganisation Cost

Any index structure that performs reorganizations has to spend extra page accesses for this task. Usually, these accesses can be performed off-line and are not immediately visible to the user. If needed, the database designer can also decide to maintain two copies of the index structure: one copy that is used for answering

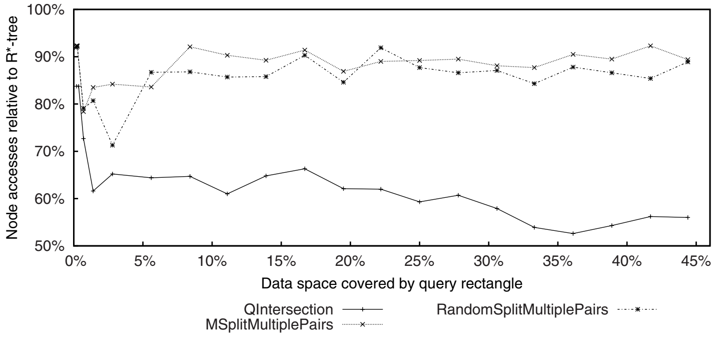
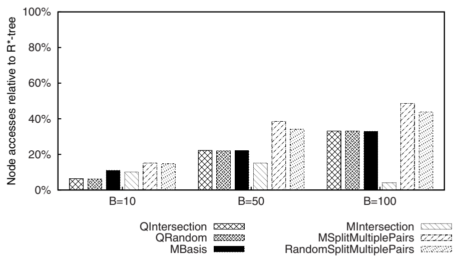
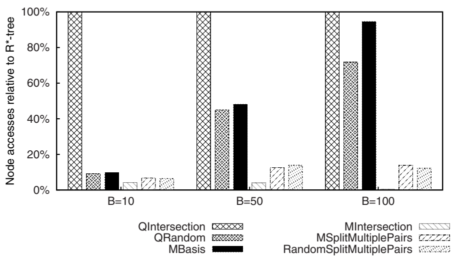


Fig. 8. Performance of proposed methods as a function of the query area



(a) Identical queries.



(b) Similar queries.

Fig. 9. Reorganization cost for Matrix-R-Trees and Query-R-Trees relative to the query cost of the R*-tree

the queries and one (shadow) copy on which reorganizations are performed. After a reorganization on the shadow copy has finished, the two structures switch roles, i.e. the reorganized structure is used for answering queries.

Nevertheless, we ran a preliminary set of experiments to examine the cost for reorganizations for the variants identified as the best structures. Note that we did not tune the reorganization parameters to the data set or query type at hand; this is beyond the scope of this paper. Fig. 9 shows that the cost for reorganization significantly depends on both the way reorganizations are handled and the type of queries. For identical queries, Query-R-Trees have a relatively low reorganizational overhead, but (especially in the non-randomized version) exhibit an extremely high reorganizational overhead for similar queries. For similar queries, on the other hand, Matrix-R-Trees perform very well, but they have inferior performance for identical queries.

5 Conclusions

Summing up the concept of query responsive index structure and the experiments presented in the previous section, we identify three main contributions of our study: Firstly, the concepts discussed in this paper describe the first type of self-organizing spatial index structures that are adapted to a database context and are general enough to be applied to a wide class of index structures. Secondly, the predominant type of queries performed using a particular index structure influences the choice of query responsiveness, especially if reorganization costs are taken into account. Finally, the experiments show that the concept of query responsive index structures is beneficial to the query performance of spatial databases and can lead to notable improvements in practice. It remains open to examine the quality of other query responsive index structures and to further investigate the exact relation between the application, the data, and the best variant of providing query responsiveness.

References

1. Agarwal, P.K., de Berg, M., Gudmundsson, J., Hammar, M., Haverkort, H.J.: Box-trees and R-trees with near-optimal query time. Discrete & Computational Geometry 28(3), 291–312 (2002)
2. Albers, S., Karpinski, M.: Randomized splay-trees. Information Processing Letters 81(4), 213–221 (2002)
3. Albers, S., Westbrook, J.: Self-organizing data structures. In: Fiat, A., Woeginger, G.J. (eds.) Dagstuhl Seminar 1996. LNCS, vol. 1442, pp. 13–51. Springer, Heidelberg (1998)
4. Arge, L.A., Barve, R.D., Hutchinson, D., Procopiuc, O., Toma, L.I., Vengroff, D.E., Wickremesinghe, R.: TPIE user manual and reference. Duke University, North Carolina (2002), http://www.cs.duke.edu/TPIE/
5. Arge, L.A., de Berg, M., Haverkort, H.J., Yi, K.: The priority R-tree: A practically efficient and worst-case optimal R-tree. In: Weikum, G., König, A.C., Deßloch, S. (eds.) Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 347–358. ACM Press, New York (2004)
6. Arge, L.A., Hinrichs, K.H., Vahrenhold, J., Vitter, J.S.: Efficient bulk operations on dynamic R-trees. Algorithmica 33(1), 104–128 (2002)
7. Arge, L.A., Procopiuc, O., Ramaswamy, S., Suel, T., Vahrenhold, J., Vitter, J.S.: A unified approach for indexed and non-indexed spatial joins. In: Zaniolo, C., Lockemann, P.C., Scholl, M.H., Grust, T. (eds.) EDBT 2000. LNCS, vol. 1777, pp. 413–429. Springer, Heidelberg (2000)
8. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-tree: An efficient and robust access method for points and rectangles. In: Garcia-Molina, H., Jagadish, H.V. (eds.) Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data. SIGMOD Record, vol. 19.2, pp. 322–331. ACM Press, New York (1990)
9. Bell, J., Gupta, G.: An evaluation of self-adjusting binary search tree techniques. Software – Practice and Experience 23(4), 369–382 (1993)
10. Berchtold, S., Keim, D.A., Kriegel, H.-P.: The X-tree: An index structure for high-dimensional data. In: Vijayaraman, T.M., Buchmann, A.P., Mohan, C., Sarda, N.L. (eds.) VLDB 1996: Proceedings of the 22nd International Conference on Very Large Data Bases, pp. 28–39. Morgan Kaufmann, San Francisco (1996)
11. Bitner, J.R.: Heuristics that dynamically organize data structures. SIAM Journal on Computing 8(1), 82–110 (1979)
12. Choubey, R., Chen, L., Rundensteiner, E.A.: In: Güting, R.H., Papadias, D., Lochovsky, F. (eds.) SSD 1999. LNCS, vol. 1651, pp. 91–108. Springer, Heidelberg (1999)
13. Cobb, M., Chung, M., Shaw, K., Arctur, D.: A self-adjusting indexing structure for spatial data. In: Proceedings of the 1995 ASPRS GIS/LIS Conference and Exhibition, vol. I, pp. 182–192 (1995)
14. Fürer, M.: Randomized splay trees. In: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 903–904. Association for Computing Machinery (1999)
15. García, Y.J., López, M.A., Leutenegger, S.T.: On optimal node splitting for R-trees. In: Gupta, A., Shmueli, O., Widom, J. (eds.) VLDB 1998: Proceedings of the 24th International Conference on Very Large Data Bases, pp. 334–344. Morgan Kaufmann, San Francisco (1998)

16. Greene, D.: An implementation and performance analysis of spatial data access methods. In: Proceedings of the Fifth International Conference on Data Engineering, pp. 606–615. IEEE Computer Society Press, Los Alamitos (1989)
17. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: Yormark, B. (ed.) SIGMOD 1984, Proceedings of Annual Meeting, pp. 47–57. ACM Press, New York (1984)
18. Kamel, I., Faloutsos, C.: Hilbert R-tree: An improved R-tree using fractals. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) Proceedings of the 20th International Conference on Very Large Data Bases (VLDB 1994), pp. 500–509. Morgan Kaufmann, San Francisco (1994)
19. Kamel, I., Khalil, M., Kouramajian, V.: Bulk insertion in dynamic R-trees. In: Kraak, M., Molenaar, M. (eds.) Proceedings of the Seventh International Symposium on Spatial Data Handling (SDH 1996), pp. 3B.31–3B.42 (1996)
20. Kanth, K.V.R., Singh, A.K.: Optimal dynamic range searching in non-replicating index structures. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 257–276. Springer, Heidelberg (1998)
21. Knuth, D.E.: Optimum binary search trees. Acta Informatica 1(1), 14–25 (1971)
22. Lomet, D.B.: Grow and post index trees: Role, techniques and future potential. In: Günther, O., Schek, H.-J. (eds.) SSD 1991. LNCS, vol. 525, pp. 183–206. Springer, Heidelberg (1991)
23. Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A.N., Theodoridis, Y.: R-trees: Theory and Applications. Springer, Berlin (2005)
24. Mehlhorn, K.: Nearly optimal binary search trees. Acta Informatica 5(4), 287–295 (1975)
25. Ross, K.A., Sitzmann, I., Stuckey, P.J.: Cost-based unbalanced R-trees. In: Proceedings of the 13th International Conference on Scientific and Statistical Database Management, pp. 202–213. IEEE Computer Society Press, Los Alamitos (2001)
26. Samet, H.: Foundations of Multidimensional and Metric Data Structures. In: The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling.Morgan Kaufmann, San Francisco (2005)
27. Sellis, T., Roussopoulos, N., Faloutsos, C.: The R^+-tree: A dynamic index for multi-dimensional objects. In: Stocker, P.M., Kent, W., Hammersley, P. (eds.) 13th International Conference on Very Large Data Bases, pp. 507–518. Morgan Kaufmann, San Francisco (1987)
28. Sleator, D.D.K., Tarjan, R.E.: Self-adjusting binary search trees. Journal of the ACM 32(3), 652–686 (1985)
29. Tao, Y., Papadias, D.: Adaptive index structures. In: CAiSE 2002 and VLDB 2002, pp. 418–429. Morgan Kaufmann, San Francisco (2002)
30. Theodoridis, Y.: The R-tree-portal (2003), http://www.rtreeportal.org
31. van den Bercken, J., Seeger, B.: An evaluation of generic bulk loading techniques. In: Apers, P.M.G., Atzeni, P., Ceri, S., Paraboschi, S., Ramamohanarao, K., Snodgrass, R.T. (eds.) VLDB 2001: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 461–470. Morgan Kaufmann, San Francisco (2001)

Refining Topological Relations between Regions Considering Their Shapes

Roland Billen[1] and Yohei Kurata[2]

[1] Geomatics Unit, University of Liege, 17 Allée du 6-Août, B-4000 Liege, Belgium
rbillen@ulg.ac.be
[2] SFB/TR 8 Spatial Cognition, Universität Bremen
Postfach 330 440, 28334 Bremen, Germany
ykurata@informatik.uni-bremen.de

Abstract. Topological relations are sometimes insufficient for differentiating spatial configurations of two objects with critical difference in their connection styles. In this paper, we present the *projective 9^+-intersection model*, which refines topological relations into *projective binary relations* by considering projective properties of the objects' shapes. This is indeed a reformulation of projective concepts of the *Dimensional Model* within the framework of the *9^+-intersection.* Thirty projective binary relations are established between two regions in $\mathbf{R}^2$, one of which has a *multi-order boundary* (region$_{+\text{mob}}$). These relations are identified computationally by applying to all theoretical relations the existing constraints for topological region-region relations and seven new specific constraints. After defining the concept of continuous neighbours between two projective binary relations, a conceptual neighbourhood graph of the 30 projective region$_{+\text{mob}}$-region relations is developed.

1 Introduction

Topological relations for categorising spatial configurations of objects have been studied extensively over the last two decades [1-15]. Especially, binary topological relations between spatial objects (point, line, regions and, to a smaller extend, bodies) are well-studied and frequently used. Several models of binary topological relations have been proposed and some of them have been adopted as OGC standards [16]. Other types of relations have also been introduced [17-20] and lots of work has still to be done to tackle the diversity and insufficiency of the representation of spatial relations. Indeed, even binary topological relations can be pushed further; for instance by considering dimension and number of topological intersections [7, 8]. However, the existing approaches cannot differentiate some spatial configurations of two objects, despite of their significant qualitative difference. For instance, Fig. 1a shows a set of spatial configurations between two bodies, which are all categorised into the same topological relation "*touch*" with a two-dimensional intersection. Similarly, Fig. 1b shows another set of configurations between a body and a line, which are all categorised into the same topological relation "*touch*" with a zero-dimensional intersection.

Distinction of such configurations is useful to perform more sophisticated spatial analyses, to perform more specific consistency checks, and so on. Such distinction based on the difference of connection styles seems particularly important when considering 3D large-scale applications; for example when categorising the spatial configurations of indoor objects (object arrangements in a room) or relationships between the components of objects (between the walls and roof of a building, between electronic components inside a computer, etc.). In such cases, considering topology only is not enough. Such applications go beyond "traditional" 2D GIS domain and could be seen closer to 3D CAD world. However, with 3D urban modelling and 3D urban GIS development, we strongly believe that the types of spatial relationships currently modelled must be extended.

Categorising further binary spatial relations implies to consider another mathematical framework. For instance, projective geometry has been adopted to categorise spatial relations in a model called the *Dimensional Model* (*DM*) [21, 22]. The motivation behind DM was to identify spatial relations between 3D objects more precisely. Although formally defined, DM suffers from a lack of standardisation and strong algebra associated with the model. DM's formal definition of spatial relations differs from the standard definition of spatial relations (e.g., the 9-intersection [2] and RCC-calculus [3]), even though DM may distinguish the same set of relations as the 9-intersection and RCC-calculus do. Thus, it has been a hard step to adopt DM in addition to other models. Furthermore, neither conceptual neighbourhood graphs nor composition tables are currently available for DM relations. This is an obstacle to performing qualitative spatial reasoning.

A recent development in the modelling methods of topological relations, the 9^+-intersection [15] has given an opportunity to reformulate DM projective concepts within a well-standardized framework of topological relations. The new combined model keeps all descriptive power of DM projective concepts and takes advantage of existing development in the 9-intersection [2] and the 9^+-intersection [15], especially in terms of qualitative spatial reasoning.

In this article, we propose a new model of refined topological relations based on projective properties of objects' shapes. First, we explain fundamental concepts of DM, such as the order of points of a spatial object (Section 2). Then, we reformulate DM with the 9^+-intersection (Section 3). We then identify all possible relations between two regions embedded in $\mathbf{R}^2$, one of which has a "projective" shape (Section 4). Then, we schematize these relations into a conceptual neighbourhood graph, whose analysis reveals some interesting properties of the relations (Section 5). And finally we conclude (Section 6).

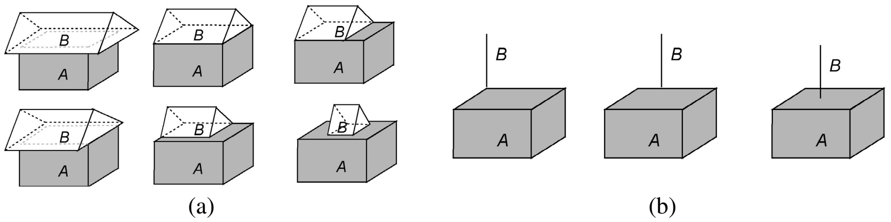


Fig. 1. Topological equivalence of (a) spatial configurations of two bodies with two-dimensional intersection and (b) those of a body and a line in zero-dimensional intersection

2 Dimensional Model (DM)

The Dimensional Model (DM) was formally defined according to affine geometry [21, 22]. It was shown later that its definition satisfies projective geometry rules [23]. This model introduced the concept of "order of each point of a convex set" to conduct the segmentation of spatial objects. New binary spatial relations between two spatial objects, called *dimensional relations*, were defined based on the connectivity of the objects' dimensional elements.

2.1 Order of Point of a Spatial Object

Every point of a closed convex set has an *order* [24]. Let C be a convex set in d-dimensional Euclidean space $\mathbf{R}^d$. The order of a point x in C, denoted $o(x,C)$, is the dimension of the intersection of all the supporting hyperplanes of C passing through x. If there is no supporting hyperplane containing x, then x is of order d. One can prove that the set of points of order d correspond exactly to the points of C's interior in the sense of Euclidean topology (not equivalent to the interior in point-set topology, as we will see later). For example, suppose a triangle in $\mathbf{R}^2$. In $\mathbf{R}^2$, an infinite number of supporting hyperplanes (in this case, lines) can pass through a vertex of the triangle (Fig. 2a). Their intersection is a subspace of dimension 0 (a point corresponding to the given vertex). Thus, this vertex is of order 0. If we take a point being located on the edge of the triangle, there is only one supporting hyperplane. This subspace is of dimension 1 and thus the order of this point is 1 (like all points of the edge other than the two vertices). Lastly, for an interior point of the triangle, there is no supporting hyperplane and thus the order of this interior point is 2 (the dimension of the embedding space). Other examples (a drop-shaped region in $\mathbf{R}^2$ and a line segment in $\mathbf{R}^2$) are presented in Figs. 2b-c.

This order concept has been extended from convex sets to topological manifolds with boundary[1], such that we can consider a wide variety of spatial objects. The definition of the extended concept is beyond the scope of this article; see [21] or [22] for a comprehensive development.

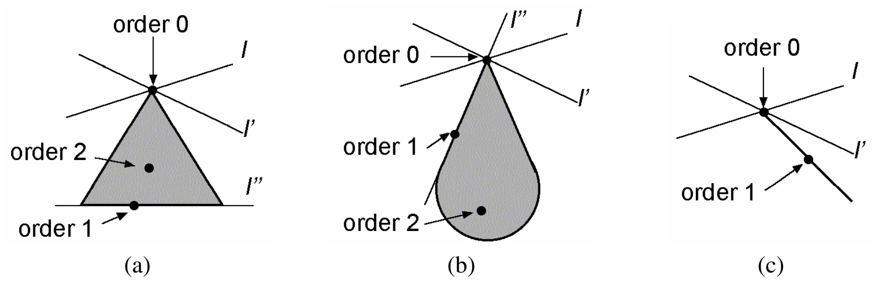


Fig. 2. Order of points of various spatial objects in $\mathbf{R}^2$

[1] Let n be a positive integer. We denote by $\mathbf{R}^n_+$ the subset of tuples $(\alpha_1,...,\alpha_n)$ with $\alpha_n \geq 0$. A subset A of $\mathbf{R}^d$ is a *topological n-manifold with boundary* if each point $x \in A$ has neighbourhood which is homeomorphic to an open subset of $\mathbf{R}^n_+$.

Strictly speaking, the object's "interior" concept in Euclidean topology, on which the original DM stands, is different from that in point-set topology [25], on which the 9-intersection stands. This difference appears when considering an n-dimensional object embedded in $\mathbf{R}^d$ ($n < d$). In this case, Euclidean topology considers that the object has only a boundary (Fig. 3a), while point-set topology considers that the object consists of a boundary (the set of two endpoints) and an interior (Fig. 3b).

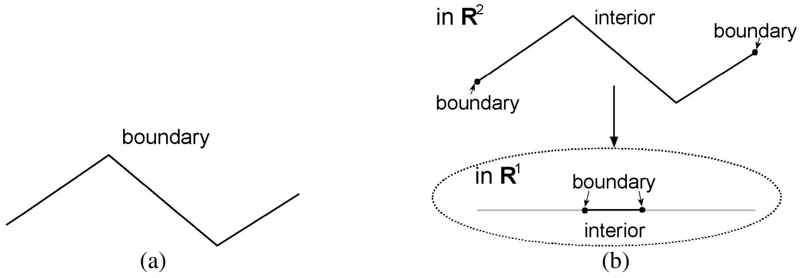


Fig. 3. Differences between Euclidean topology and point-set topology for n-dimensional object embedded in $\mathbf{R}^d$ ($n<d$)

To make DM consistent with the models based on point-set topology, the definition of point order has been extended [21], such that the previous definition of order is applied only to the points on the object's boundary (in the sense of point-set topology), while we consider that the points in object's interior (in the sense of point-set topology) has the same order with the object's dimension. In the previous definition of point order, for example, points on a line can be of order 0 or 1 and the points of order 0 are not only line extremities (Fig. 4a). In the extended definition of point order, the line's "extremities" has order 0 while the line's "interior" has order 1 (Fig. 4b).

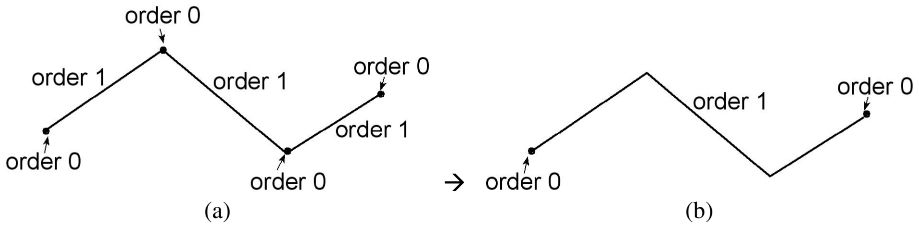


Fig. 4. The extension of point order concept for point-set objects

Another issue occurs when applying the previous order definition to the inflexion points. In this case, point's order is higher than our intuition; i.e. equal to the order of interior points (Fig. 5a). Thus, in the extended definition of point order, if an object of dimension d has a boundary point of order d, the order of this point is reduced to d-1 (Fig. 5b).

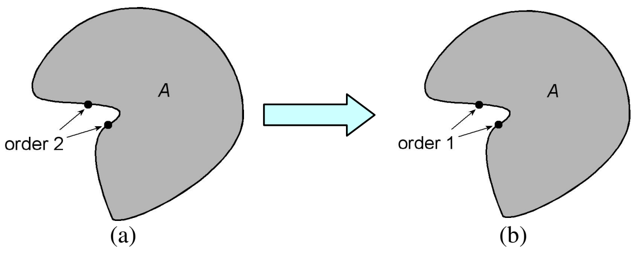


Fig. 5. Modification of inflexion points' order

This extended point order concept enables us to consider the segmentation of object's boundary and consequently to refine binary relationships between objects by considering the connectivity of objects "topological" primitives. It has given birth to the DM model through the concepts of dimensional elements and dimensional relationships.

2.2 Dimensional Elements and Dimensional Relationships

Based on the point order concept, we can consider the subsets of spatial objects, called *dimensional elements*. Dimensional elements are formally defined as follow:

- The *n*-dimensional element (called *nD-element*) of a *d*-dimensional spatial object *C* ($n \le d$) corresponds to the set of all of *C*'s points (or parts) of order 0 to *n*.
- The *n*D-element of a spatial object *C* has an *extension* and may have a *limit*. The extension is the subset of *C* formed by its points of order *n*, and the limit is the subset of *C* formed by its points of order 0 to order (*n*-1).

Thus, if the *n*D-element has a limit, this limit corresponds to (*n*-1)D-element. The 0D-element does not have a limit by definition. For instance, let us consider a polygon in Fig. 6a. This polygon is composed by points of order 0, 1, and 2. Thus, we can consider 0D-, 1D-, and 2D-elements for this convex. The different extension and limits are presented in the Figs. 6b-d. It should be noted that each object has one and only one *n*D-element. Thus, *n*D-element may be disconnected (e.g., a polygon's 0D element)

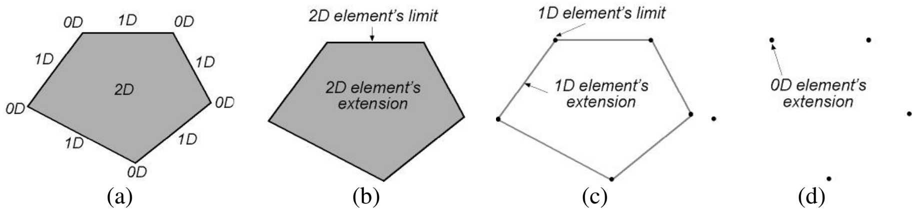


Fig. 6. (a) Order of points and (b-d) dimensional elements of a polygon

Dimensional relationships between two spatial objects are spatial relationships determined by the connectivity of their dimensional elements. We consider three connection styles, namely *total relation*, *partial relation* and *no relation* (*non-existent*), instead of presence or absence of intersections (Fig. 7).

- A dimensional element is in *total relation* with another dimensional element if their intersection is equal to the first element, and if the intersection between their extensions is not empty.
- A dimensional element is in *partial relation* with another dimensional element if their intersection is not equal to the first element, and if the intersection between their extensions is not empty.
- A dimensional element is in *no relation* (*non-existent*) with another dimensional element if the intersection between their extensions is empty.

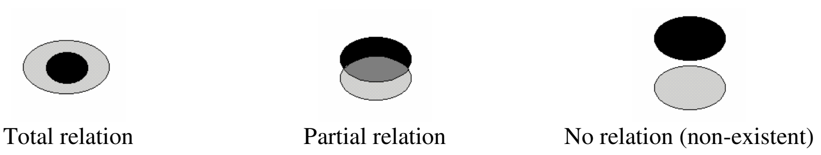
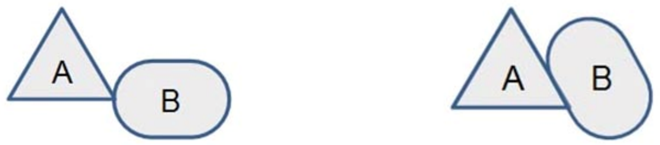


Fig. 7. Three types of connection styles between two 2D-elements (from black element to grey element)

2.3 Categorising Spatial Relationships Using Dimensional Relationships

The spatial relationship between two objects can be expressed by the set of connection styles between pairs of dimensional elements of these two objects. For example, let us consider a triangle *A* (with 2D-, 1D-, and 0D-elements) and an ellipse *B* (with 2D- and 1D-elements, but no 0D-element) (Fig. 8). The dimensional relationships between two objects *A* and *B* are determined in the following sequence: first, check the dimensional relationship between *A*'s 2D-element and each of *B*'s dimensional elements; then, check the dimensional relationship between *A*'s 1D-element and each of *B*'s dimensional elements, and so on. The dimensional relationships between *B* and *A* can be determined by the same approach. A dimensional relationship is coded as *RnDm*, where *R* stands for relation, *nD* for the dimension of the element of the first object, and *m* for the dimension of the element of the second object. For instance, R2D1 represents the dimensional relationships between the 2D-element of the first object and the 1D-element of the second object.



R2D2=non-existent	R2D1=non-existent	R2D2=non-existent	R2D1=non-existent
R1D2=non-existent	R1D1=non-existent	R1D2=non-existent	R1D1=partial
R0D2=non-existent	R0D1=partial	R0D2=non-existent	R0D1=partial

Fig. 8. Two examples of categorization of spatial relationships using dimensional relationships

3 Reformulating the Dimensional Model with the 9^+-Intersection

The Dimensional Model (DM) presented in the previous section has a strong correspondence with the *9-intersection* [2]. The 9-intersection and its extensions have been frequently adopted in the studies of topological relations (e.g., [2, 10-13, 15]). Based on point-set topology [25], this model distinguishes the *interior*, *boundary*, and *exterior* of each spatial object, which are also called the object's *topological parts*. The topological relation between two spatial objects A and B is characterized by the intersections of A's three topological parts and B's three topological parts, which are concisely represented by the *9-intersection matrix* in Eqn. 1. A°, ∂A, and A^- are A's interior, boundary, and exterior, while B°, ∂B, and B^- are B's interior, boundary, and exterior, respectively. Normally, topological relations are distinguished by the presence or absence of these nine types of intersections.

$$M(A,B)=\begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} \tag{1}$$

In DM, as presented in Section 2, the points that form a spatial object are distinguished by their orders. Among the points that form an n-dimensional spatial object, the points of order n form the object's interior, while the points of order 0 to n-1 form the object's boundary. This implies the presence of a certain correspondence between DM and the 9-intersection. Meanwhile, if $n \geq 2$, DM considers n subsets of the boundary, whereas the 9-intersection cannot distinguish these subsets of the boundary. Instead, the *9^+-intersection* [15] enables such distinction of boundary subsets within the framework of the 9-intersection.

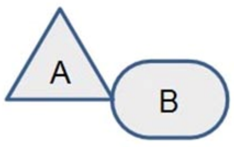
The *9^+-intersection* supports the subdivision of objects' topological parts by nesting the 9-intersection matrix. For instance, the topological relation between a directed line segment D and a simple region R is captured by the *9^+-intersection matrix* in Eqn. 2, as D's boundary consists of two subparts (start point $\partial_s D$ and end point $\partial_e D$). The support of such subdivision is useful when a certain topological parts consist of multiple subsets that are qualitatively different. In our case, the boundaries of two spatial objects A and B can be distinguished into multiple subsets, each of which consists of the points of a specific order. Therefore, the relation between two spatial objects A and B is captured by the 9^+-intersection matrix in Eqn. 3, where n_A is A's dimension, and n_B is B's dimension, $\partial_i A$ is the set of points of order i in A $(0 \leq i \leq n_A{-}1)$, and $\partial_j B$ is the set of points of order j in B $(0 \leq j \leq n_B{-}1)$. Similarly, if we distinguish A's boundary subset, but not B's boundary subset, then the relation between A and B is captured by the 9^+-intersection in Eqn. 4.

$$M^+(D,R)=\begin{pmatrix} D^\circ \cap R^\circ & D^\circ \cap \partial R & D^\circ \cap R^- \\ \begin{bmatrix} \partial_s D \cap R^\circ \\ \partial_e D \cap R^\circ \end{bmatrix} & \begin{bmatrix} \partial_s D \cap \partial R \\ \partial_e D \cap \partial R \end{bmatrix} & \begin{bmatrix} \partial_s D \cap R^- \\ \partial_e D \cap R^- \end{bmatrix} \\ D^- \cap R^\circ & D^- \cap \partial R & D^- \cap R^- \end{pmatrix} \tag{2}$$

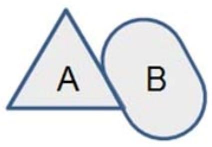
$$M^{+}(A,B)=\begin{pmatrix} A^{\circ}\cap B^{\circ} & \left[A^{\circ}\cap\partial_{n_B-1}B \ \cdots \ A^{\circ}\cap\partial_0 B\right] & A^{\circ}\cap B^{-} \\ \begin{bmatrix}\partial_{n_A-1}A\cap B^{\circ}\\ \vdots \\ \partial_0 A\cap B^{\circ}\end{bmatrix} & \begin{bmatrix}\partial_{n_A-1}A\cap\partial_{n_B-1}B & \cdots & \partial_{n_A-1}A\cap\partial_0 B\\ \vdots & \ddots & \vdots \\ \partial_0 A\cap\partial_{n_B-1}B & \cdots & \partial_0 A\cap\partial_0 B\end{bmatrix} & \begin{bmatrix}\partial_{n_A-1}A\cap B^{-}\\ \vdots \\ \partial_0 A\cap B^{-}\end{bmatrix} \\ A^{-}\cap B^{\circ} & \left[A^{-}\cap\partial_{n_B-1}B \ \cdots \ A^{-}\cap\partial_0 B\right] & D_1^{-}\cap B^{-}\end{pmatrix} \tag{3}$$

$$M^{+}(A,B)=\begin{pmatrix} A^{\circ}\cap B^{\circ} & A^{\circ}\cap\partial B & A^{\circ}\cap B^{-} \\ \begin{bmatrix}\partial_{n-1}A\cap B^{\circ}\\ \vdots \\ \partial_0 A\cap B^{\circ}\end{bmatrix} & \begin{bmatrix}\partial_{n-1}A\cap\partial B\\ \vdots \\ \partial_0 A\cap\partial B\end{bmatrix} & \begin{bmatrix}\partial_{n-1}A\cap B^{-}\\ \vdots \\ \partial_0 A\cap B^{-}\end{bmatrix} \\ A^{-}\cap B^{\circ} & A^{-}\cap\partial B & D_1^{-}\cap B^{-}\end{pmatrix} \tag{4}$$

Precisely speaking, the captured relations are no longer topological relations when $n \geq 2$, but their refinements, because the order of a point is not invariant under topological transformation. Therefore, the relations distinguished by the 9^{+}-intersection matrix in Eqn. 3 or 4 are named *binary projective relations*. In addition, the model of spatial relations based on the 9^{+}-intersection matrix in Eqn. 3 or 4 is called the *projective 9^{+}-intersection.* As examples, Fig. 9 shows how the projective 9^{+}-intersection captures projective relations between a triangle and an eclipse. We also call an object whose boundary consists of points of different orders (e.g., polygons) *the object with a multi-order boundary*, or in short, *objects*$_{+mob}$.



$$\begin{pmatrix} \phi & \phi & \neg\phi \\ \begin{bmatrix}\phi\\ \phi\end{bmatrix} & \begin{bmatrix}\phi\\ \neg\phi\end{bmatrix} & \begin{bmatrix}\neg\phi\\ \neg\phi\end{bmatrix} \\ \neg\phi & \neg\phi & \neg\phi\end{pmatrix}$$



$$\begin{pmatrix} \phi & \phi & \neg\phi \\ \begin{bmatrix}\phi\\ \phi\end{bmatrix} & \begin{bmatrix}\neg\phi\\ \neg\phi\end{bmatrix} & \begin{bmatrix}\neg\phi\\ \neg\phi\end{bmatrix} \\ \neg\phi & \neg\phi & \neg\phi\end{pmatrix}$$

Fig. 9. Two examples of categorization of spatial relations using the projective 9^{+}-intersection

The projective 9^{+}-intersection is upward compatible with DM, because, given a 9^{+}-intersection matrix in Eqn. 3, we can uniquely determine the dimensional relation in DM (i.e., a complete set of R*n*D*m* expressions) through the conversion formula in Eqn. 5 (compare Figs. 8-9). Meanwhile, the new model and DM has the following differences:

- DM considers the intersections with respect to *k*-dimensional element (i.e., the set of points of order 1 to *k*), while the new model considers the intersections with respect to the extension of *k*-dimensional element (i.e., the set of points of order *k*).

- The new model also considers the intersections with respect to the objects' exteriors.
- DM distinguishes three styles of intersections (i.e., partial, total, and non-existent), while the new model distinguishes only two styles (i.e., X intersects with Y or not). This change does not reduce the model's representation capability, because it becomes possible to determine whether X overlaps with Y (partial relation), X is included in Y (total relation), or else (no relation), considering the intersections concerning the objects' exteriors.

$$\begin{array}{ll}
A_{O(A)} = A^{\circ},\ A_i = \partial_i A \left(0 \le i < O(A) - 1\right) & \\
B_{O(B)} = B^{\circ},\ B_i = \partial_i B \left(0 \le i < O(B) - 1\right) & \\
A_n \cap B_m \ne \phi \quad \wedge \quad A_n \cap \left(\bigcup_{k=n+1}^{O(B)} B_k \cup B^- \right) \ne \phi & \rightarrow \quad \mathrm{R}n\mathrm{D}m = \mathit{partial} \\
A_n \cap B_m \ne \phi \quad \wedge \quad A_n \cap \left(\bigcup_{k=n+1}^{O(B)} B_k \cup B^- \right) = \phi & \rightarrow \quad \mathrm{R}n\mathrm{D}m = \mathit{total} \\
A_n \cap B_m = \phi & \rightarrow \quad \mathrm{R}n\mathrm{D}m = \mathit{non\text{-}existent}
\end{array} \tag{5}$$

The projective 9^+-intersection serves as a refinement of the 9-intersection, as easily expected from the shape of the underlying matrix. This means that spatial relations under the new model can be systematically categorized based on the sets of topological relations identified in the previous studies (e.g., [2]), as well as that the spatial arrangement of two objects that falls into the same topological relations in the previous relation set may be distinguished by the new model. The next sections discuss such characteristics of the new model as a refinement of the 9-intersection.

4 Projective Relations between a Region with Multi-order Boundary and a Region

There are in theory $2^{4\times3} = 4096$ projective relations between a region with multi-order boundary and a region in $\mathbf{R}^2$, because the relations are represented by the 9^+-intersection matrix with 4x3 two-valued elements (e.g., Fig. 9). However, only a small subset of these relations can be realized in a particular space. As a refinement of the 9-intersection, all the constraints of the 9-intersection for topological region-region relations, identified in [2], can be applied to the projective 9^+-intersection. In addition, due to the subdivision of the region's boundary, the following seven specific constraints are additionally applied, where the set of points of order n in the boundary of a spatial object X is called *X's order n boundary*:

- **Constraint 1.** If A's order 0 boundary intersects with B's interior, then A's order 1 boundary must intersect with B's interior
- **Constraint 2.** If A's order 0 boundary intersects with B's exterior, then A's order 1 boundary must intersect with B's exterior
- **Constraint 3.** If A's order 0 boundary intersects with B's interior and B's exterior, then A's boundary must intersect with B's boundary
- **Constraint 4.** If A's order 1 boundary intersects only with B's exterior, then A's order 0 boundary must not intersect with B's interior
- **Constraint 5.** If A's order 1 boundary intersects only with B's interior, then A's order 0 boundary must not intersect with B's exterior

- **Constraint 6.** *A*'s order 1 boundary intersects with at least one part of *B*
- **Constraint 7.** *A*'s order 0 boundary intersects with at least one part of *B*

Fig. 10 illustrates these constraints using the 9^{+}-intersection matrix.

$$M^{+}(A,B) \neq \begin{pmatrix} - & - & - \\ \begin{bmatrix} \phi \\ \neg\phi \end{bmatrix} & \begin{bmatrix} - \\ - \end{bmatrix} & \begin{bmatrix} - \\ - \end{bmatrix} \\ - & - & - \end{pmatrix}$$

Constraint 1

$$M^{+}(A,B) \neq \begin{pmatrix} - & - & - \\ \begin{bmatrix} - \\ - \end{bmatrix} & \begin{bmatrix} - \\ - \end{bmatrix} & \begin{bmatrix} \phi \\ \neg\phi \end{bmatrix} \\ - & - & - \end{pmatrix}$$

Constraint 2

$$M^{+}(A,B) \neq \begin{pmatrix} - & - & - \\ \begin{bmatrix} - \\ \neg\phi \end{bmatrix} & \begin{bmatrix} \phi \\ \phi \end{bmatrix} & \begin{bmatrix} - \\ \neg\phi \end{bmatrix} \\ - & - & - \end{pmatrix}$$

Constraint 3

$$M^{+}(A,B) \neq \begin{pmatrix} - & - & - \\ \begin{bmatrix} \phi \\ \neg\phi \end{bmatrix} & \begin{bmatrix} \phi \\ - \end{bmatrix} & \begin{bmatrix} \neg\phi \\ - \end{bmatrix} \\ - & - & - \end{pmatrix}$$

Constraint 4

$$M^{+}(A,B) \neq \begin{pmatrix} - & - & - \\ \begin{bmatrix} \neg\phi \\ - \end{bmatrix} & \begin{bmatrix} \phi \\ - \end{bmatrix} & \begin{bmatrix} \phi \\ \neg\phi \end{bmatrix} \\ - & - & - \end{pmatrix}$$

Constraint 5

$$M^{+}(A,B) \neq \begin{pmatrix} - & - & - \\ \begin{bmatrix} \phi \\ - \end{bmatrix} & \begin{bmatrix} \phi \\ - \end{bmatrix} & \begin{bmatrix} \phi \\ - \end{bmatrix} \\ - & - & - \end{pmatrix}$$

Constraint 6

$$M^{+}(A,B) \neq \begin{pmatrix} - & - & - \\ \begin{bmatrix} - \\ \phi \end{bmatrix} & \begin{bmatrix} - \\ \phi \end{bmatrix} & \begin{bmatrix} - \\ \phi \end{bmatrix} \\ - & - & - \end{pmatrix}$$

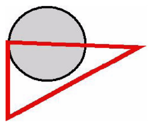
Constraint 7

Fig. 10. New constraints applied to the matrix of the projective 9^{+}-intersection

By applying these constraints, the number of relations is reduced from 4096 to 30 (Figs. 11-12). All of these 30 relations normally have geometric realization in $\mathbf{R}^2$. Exceptionally, when *A*'s order 0 boundary consist of three distinctive subparts (i.e., when *A* is a triangle), the relation "*overlap-11*" is impossible. These 30 relationships are indeed a refinement of eight topological relations between regions. Consequently, we have decided to keep the same names (i.e. *overlap*, *contains*, *inside*, *covers*, *coveredBy*, *equal*, *meet*, and *disjoint* [4]) and just added a number to them which reflects the number of refined cases.

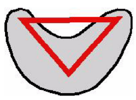
5 Conceptual Neighbourhood Graph

The 30 projective relations derived in the previous section are organized into a similarity-based schema, called a *conceptual neighbourhood graph* [26]. Conceptual neighbourhood graphs are frequently used for schematizing, analyzing, and visualizing a set of spatial relations [4, 9, 11, 14, 15, 19, 26-30]. In a conceptual neighbourhood graph, each node corresponds to a spatial relation, and two nodes are linked if their corresponding relations are *conceptual neighbours*. Different definitions of conceptual neighbours lead to different graph shapes for the same set of relations [26, 28]. This paper considers that two projective relations between a region with a multi-order boundary ($\text{region}_{+\text{mob}}$) *A* and a region *B* are conceptual neighbours if there is a configuration of one relation that can switch to a configuration of another relation by transforming *A* continuously, such that one vertex or edge loses/gains one intersection with *B*'s interior, boundary, or exterior. Accordingly, the transformation in Fig. 13a



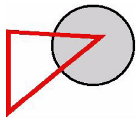
	A°	dA	A-
B°	1	1	1
dB1	1	1	1
dB0	0	1	1
B-	1	1	1

overlap - 1



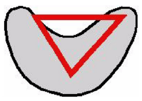
	A°	dA	A-
B°	1	1	1
dB1	1	1	1
dB0	1	0	0
B-	1	1	1

overlap - 2



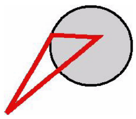
	A°	dA	A-
B°	1	1	1
dB1	1	1	1
dB0	1	0	1
B-	1	1	1

overlap - 3



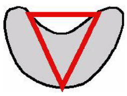
	A°	dA	A-
B°	1	1	1
dB1	1	1	1
dB0	1	1	0
B-	1	1	1

overlap - 4



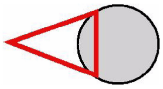
	A°	dA	A-
B°	1	1	1
dB1	1	1	1
dB0	1	1	1
B-	1	1	1

overlap - 5



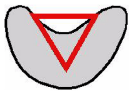
	A°	dA	A-
B°	1	1	1
dB1	1	0	1
dB0	0	1	0
B-	1	1	1

overlap - 6



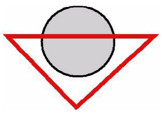
	A°	dA	A-
B°	1	1	1
dB1	1	0	1
dB0	0	1	1
B-	1	1	1

overlap - 7



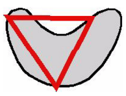
	A°	dA	A-
B°	1	1	1
dB1	1	0	1
dB0	1	1	0
B-	1	1	1

overlap - 8



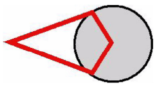
	A°	dA	A-
B°	1	1	1
dB1	1	1	1
dB0	0	0	1
B-	1	1	1

overlap - 9



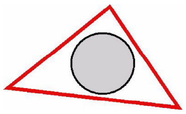
	A°	dA	A-
B°	1	1	1
dB1	1	1	1
dB0	0	1	0
B-	1	1	1

overlap - 10



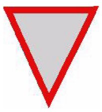
	A°	dA	A-
B°	1	1	1
dB1	1	0	1
dB0	1	1	1
B-	1	1	1

overlap - 11



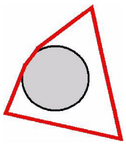
	A°	dA	A-
B°	1	1	1
dB1	0	0	1
dB0	0	0	1
B-	0	0	1

contains



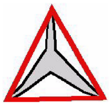
	A°	dA	A-
B°	1	0	0
dB1	0	1	0
dB0	0	1	0
B-	0	0	1

equal



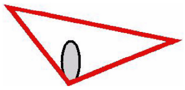
	A°	dA	A-
B°	1	1	1
dB1	0	1	1
dB0	0	1	1
B-	0	0	1

covers - 1



	A°	dA	A-
B°	1	1	1
dB1	0	0	1
dB0	0	1	0
B-	0	0	1

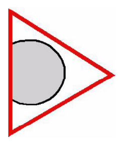
covers - 2



	A°	dA	A-
B°	1	1	1
dB1	0	0	1
dB0	0	1	1
B-	0	0	1

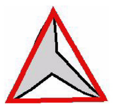
covers - 3

Fig. 11. Thirty possible projective relations between a region with a multi-order boundary and a region in **R²** – Part 1



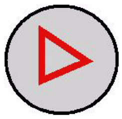
	A°	dA	A-
B°	1	1	1
dB1	0	1	1
dB0	0	0	1
B-	0	0	1

covers - 4



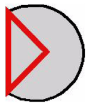
	A°	dA	A-
B°	1	1	1
dB1	0	1	1
dB0	0	1	0
B-	0	0	1

covers - 5



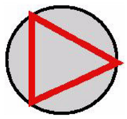
	A°	dA	A-
B°	1	0	0
dB1	1	0	0
dB0	1	0	0
B-	1	1	1

inside



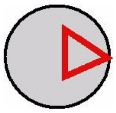
	A°	dA	A-
B°	1	0	0
dB1	1	1	0
dB0	1	1	0
B-	1	1	1

coveredBy - 1



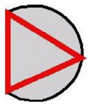
	A°	dA	A-
B°	1	0	0
dB1	1	0	0
dB0	0	1	0
B-	1	1	1

coveredBy - 2



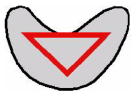
	A°	dA	A-
B°	1	0	0
dB1	1	0	0
dB0	1	1	0
B-	1	1	1

coveredBy - 3



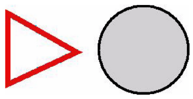
	A°	dA	A-
B°	1	0	0
dB1	1	1	0
dB0	0	1	0
B-	1	1	1

coveredBy - 4



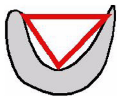
	A°	dA	A-
B°	1	0	0
dB1	1	1	0
dB0	1	0	0
B-	1	1	1

coveredBy - 5



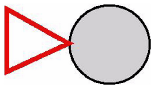
	A°	dA	A-
B°	0	0	1
dB1	0	0	1
dB0	0	0	1
B-	1	1	1

disjoint



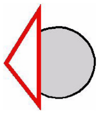
	A°	dA	A-
B°	0	0	1
dB1	0	0	1
dB0	0	1	0
B-	1	1	1

meet - 1



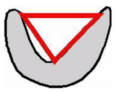
	A°	dA	A-
B°	0	0	1
dB1	0	0	1
dB0	0	1	1
B-	1	1	1

meet - 2



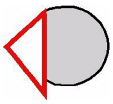
	A°	dA	A-
B°	0	0	1
dB1	0	1	1
dB0	0	0	1
B-	1	1	1

meet - 3



	A°	dA	A-
B°	0	0	1
dB1	0	1	1
dB0	0	1	0
B-	1	1	1

meet - 4



	A°	dA	A-
B°	0	0	1
dB1	0	1	1
dB0	0	1	1
B-	1	1	1

meet - 5

Fig. 12. Thirty possible projective relations between a region with a multi-order boundary and a region in $\mathbf{R}^2$ – Part 2

establishes continuous neighbours, but that in Fig. 13b, which yields the generation of three intersections ($\partial_0 A \cap B^\circ$, $\partial_1 A \cap B^\circ$, and $\partial_1 A \cap \partial B$) and loss of one intersection ($\partial_0 A \cap \partial B$), is not accepted. Acceptable continuous transformation always results in the change of one of six elements in the middle two rows of the matrix (i.e., $\partial_0 A \cap B^\circ$, $\partial_0 A \cap \partial B$, $\partial_0 A \cap B^-$, $\partial_1 A \cap B^\circ$, $\partial_1 A \cap \partial B$, and $\partial_1 A \cap B^-$). Thus, all candidates for conceptual neighbours are derived computationally from the patterns of the 9^+-intersection matrix that correspond to the 30 projective relations.

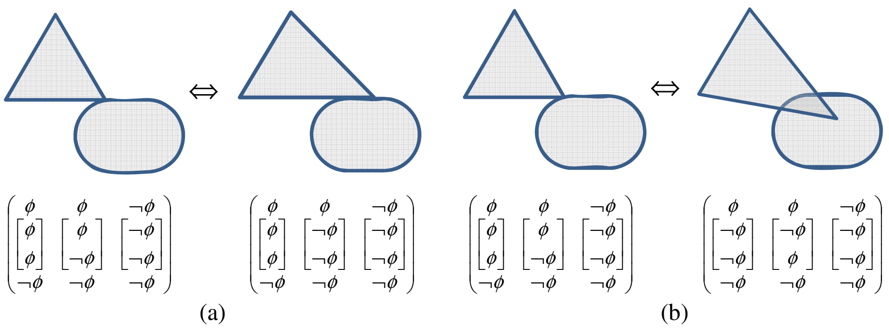


Fig. 13. (a) A continuous transformation that establishes conceptual neighbours and (b) another continuous transformation that does not establish conceptual neighbours

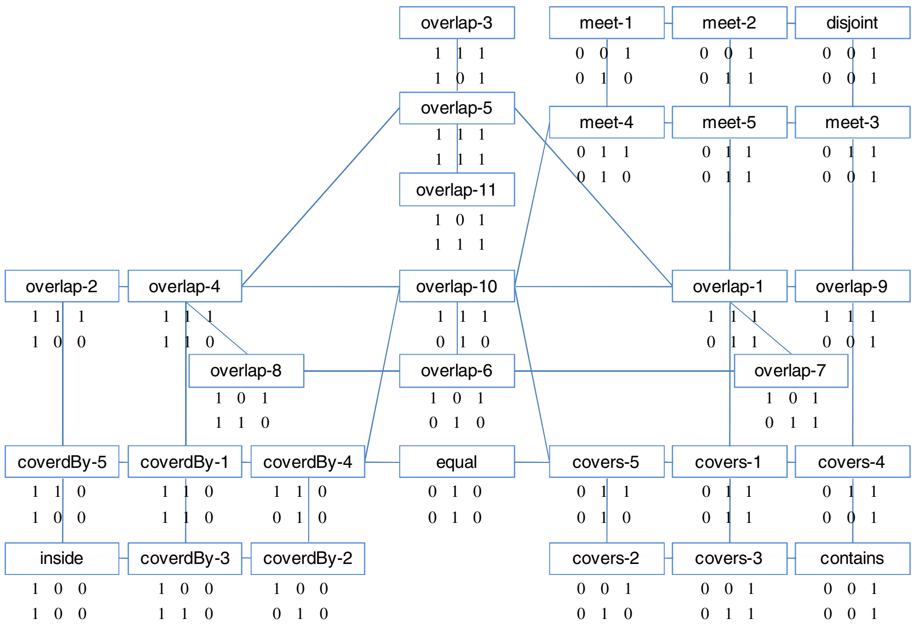
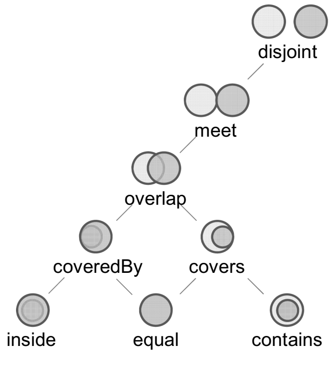
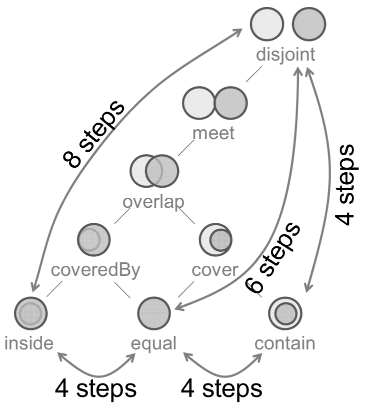


Fig. 14. Conceptual neighbourhood graphs of the 30 projective relations between a region with a multi-order boundary and a region in R², together with the six elements in the matrix's middle two rows

Based on the previous definition of conceptual neighbours, we developed the conceptual neighbourhood graph of the 30 projective region$_{+\text{mob}}$-region relations (Fig. 14). Basically, the developed graph has both horizontal and vertical symmetry axes, even though the upper-left part is missing and overlap-refinements do not follow the horizontal symmetric axis. The spatial arrangement of the 30 projective region$_{+\text{mob}}$-region relations in this conceptual neighbourhood graph corresponds to the arrangement of topological region-region relations in their conceptual neighbourhood graph [4] (Fig. 15a). This highlights the characteristics of these 30 projective region$_{+\text{mob}}$-region relations as the refinement of topological region-region relations. The comparison of two graphs shows that the number of steps from one relation to another relation is larger in our graph, except those from *disjoint* to *contains* and vice versa (Fig. 15b).



(a)

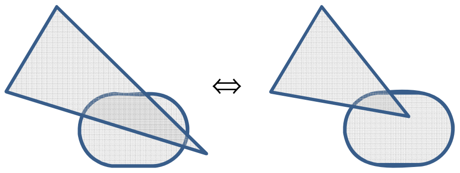


(b)

Fig. 15. (a) Conceptual neighbourhood graphs of eight topological region-region relations [4] and (b) numbers of steps between nodes in the corresponding conceptual neighbourhood graph of projective region$_{+\text{mob}}$-region relations in Fig. 14

Through the analysis of the developed conceptual neighbourhood graph, we found the following facts:

- The pairs of relations with one difference in the matrices' middle two rows, where one has $(\neg\phi \quad \phi \quad \neg\phi)$ and another has $(\phi \quad \phi \quad \neg\phi)$ or $(\neg\phi \quad \phi \quad \phi)$ with respect to one of these two rows (e.g., overlap-3 and overlap-9), are not conceptual neighbours, because the shift between these two relations require that one vertex or one edge of the triangle skips over the region's boundary (Fig. 16).
- The matrices' middle two rows of the relations on the graph's vertical centre line (i.e., overlap-3, -5, -6, -10, -11, and equal) are left-right symmetric.
- In each pair of relations located symmetrically with respect to the graph's vertical centre line (e.g., inside and contain), the matrix of one relation is derived from the matrix of another relation by exchanging the matrix's elements with respect to B° and B^{-}. This means that one relation is derived from another relation by reversing B's interior and exterior, assuming a spherical surface that embeds the configurations of the relations.



$$\begin{pmatrix} \phi & \phi & \neg\phi \\ \begin{bmatrix} \neg\phi \\ \phi \end{bmatrix} & \begin{bmatrix} \neg\phi \\ \phi \end{bmatrix} & \begin{bmatrix} \neg\phi \\ \neg\phi \end{bmatrix} \\ \neg\phi & \neg\phi & \neg\phi \end{pmatrix} \qquad \begin{pmatrix} \phi & \phi & \neg\phi \\ \begin{bmatrix} \neg\phi \\ \neg\phi \end{bmatrix} & \begin{bmatrix} \neg\phi \\ \phi \end{bmatrix} & \begin{bmatrix} \neg\phi \\ \neg\phi \end{bmatrix} \\ \neg\phi & \neg\phi & \neg\phi \end{pmatrix}$$

Fig. 16. Non-continuous transformation, even though that yields only one change in the middle two rows in the 9^+-intersection matrix

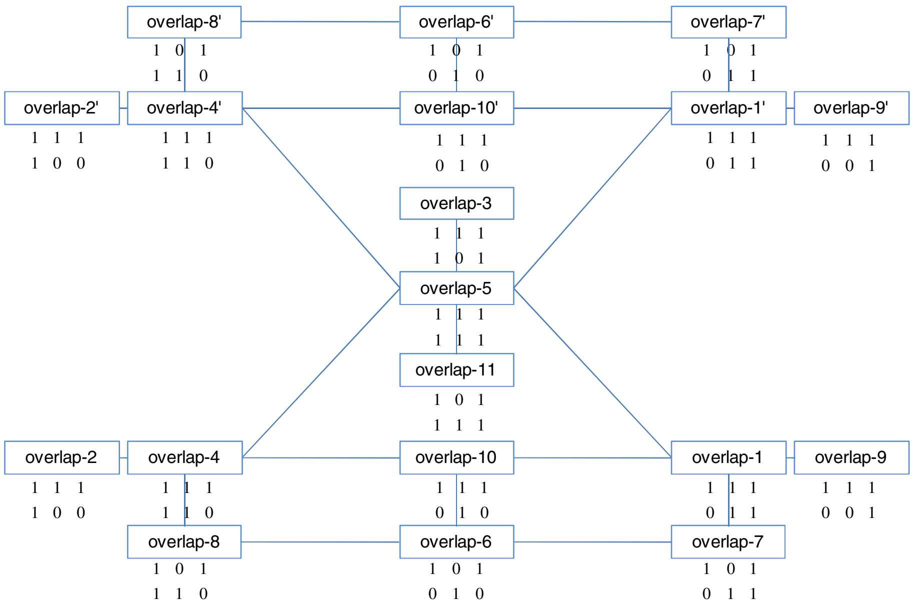


Fig. 17. Virtual symmetry of the conceptual neighbourhood graph in Fig. 3 with respect to overlap refinements. Overlap-1' to overlap-10' have no geometric realization, however.

- Similarly, in each pair of relations located symmetrically with respect to the graph's horizontal centre line (e.g., disjoint and contain), the matrix of one relation is derived from the matrix of another relation by exchanging the matrix's elements with respect to A° and A^-. This means that one relation is derived from another relation by reversing A's interior and exterior, assuming a spherical surface that embeds the configurations of the relations.
- Overlap-1 to overlap-11 follow the vertical symmetric axis only. We can consider a virtual graph that is horizontally and vertically symmetric (Fig. 17), considering the patterns of their projective 9^+-intersection matrices. In this graph, however, overlap-1' to overlap-10' have no geometric realization.

From the third and fourth findings, we can expect that the upper-left empty part of the conceptual neighbourhood graph in Fig. 14 potentially domiciles the relations that appear in a spherical surface, but not in a plane. This is analogous to Egenhofer's [11] finding of three additional region-region relations on a spherical surface, namely *attaches*, *embraces*, and *entwined.* Thus, it is expected that the relations at the graph's upper-left empty part neighbour should be the refinements of attaches, embraces, and entwined relations.

6 Conclusions

This paper developed a new model of projective relations based on the Dimensional Model [22] and the 9^+-intersection [15]. As a first step, this paper identified 30 projective relations between a region with a multi-order boundary and a region (i.e., projective $\text{region}_{+\text{mob}}$-region relations) in $\mathbf{R}^2$, and schematized these 30 relations into a conceptual neighbourhood graph. Naturally, the next target is the relations between two regions, both with a multi-order boundary (i.e., projective $\text{region}_{+\text{mob}}$- $\text{region}_{+\text{mob}}$ relations). By replacing regions by regions with multi-order boundaries, the number of projective relations will increase, and the conceptual neighbourhood graph will become more complicated (probably more high-dimensional). It is also expected that the shapes of regions with multi-order boundaries influence the realizability of some projective relations. For instance, a triangle and a quadrilateral cannot have an *equal* relation.

Another challenge is to identify and analyze projective relations in a three dimensional space. Systematic analyses of three-dimensional topological relations were conducted in [10, 21], identifying many relations that cannot be realized in $\mathbf{R}^2$. Similar analysis should be possible for projective relations.

An interesting and promising topic is the *composition* of projective relations. The composition of spatial relations is an operation that derives possible relations between two spatial objects from the knowledge of the relations between each of these objects and the common third object. Such compositions of spatial relations have been frequently discussed as a foundation of qualitative spatial reasoning [6, 11, 14, 18, 27-29, 31, 32]. It is an interesting question whether the compositions of projective relations become crisper than those of topological relations, as the projective relations consider the dimensional characteristics of the regions' boundaries.

References

1. Egenhofer, M., Franzosa, R.: Point-Set Topological Spatial Relations. International Journal of Geographical Information Systems 5, 161–174 (1991)
2. Egenhofer, M., Herring, J.: Categorizing Binary Topological Relationships between Regions, Lines and Points in Geographic Databases. In: Egenhofer, M., Herring, J., Smith, T., Park, K. (eds.): NCGIA Technical Reports 91-7. National Center for Geographic Information and Analysis, Santa Barbara, CA, USA (1991)
3. Randell, D., Cui, Z., Cohn, A.: A Spatial Logic Based on Regions and Connection. In: Nebel, B., Rich, C., Swarout, W. (eds.) 3rd International Conference on Knowledge Representation and Reasonping, pp. 165–176. Morgan Kaufmann, San Francisco (1992)

4. Egenhofer, M., Al-Taha, K.: Reasoning about Gradual Changes of Topological Relationships. In: Frank, A.U., Formentini, U., Campari, I. (eds.) GIS 1992. LNCS, vol. 639, pp. 196–219. Springer, Heidelberg (1992)
5. Clementini, E., Di Felice, P., van Oosterom, P.: A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In: Abel, D., Ooi, B.C. (eds.) SSD 1993. LNCS, vol. 692, pp. 277–295. Springer, Heidelberg (1993)
6. Egenhofer, M.: Deriving the Composition of Binary Topological Relations. Journal of Visual Languages and Computing 5, 133–149 (1994)
7. Egenhofer, M., Franzosa, R.: On the Equivalence of Topological Relations. International Journal of Geographical Information Systems 9, 133–152 (1995)
8. Clementini, E., Di Felice, P.: Topological Invariants for Lines. IEEE Transactions on Knowledge and Data Engineering 10, 38–54 (1998)
9. Hornsby, K., Egenhofer, M., Hayes, P.: Modeling Cyclic Change. In: Chen, P., Embley, D., Kouloumdjian, J., Liddle, S., Roddick, J. (eds.) TABLEAUX 1997. LNCS, vol. 1227, pp. 98–109. Springer, Heidelberg (1999)
10. Zlatanova, S.: On 3D Topological Relationships. In: 11th International Workshop on Database and Expert Systems Applications, pp. 913–924. IEEE Computer Society, Los Alamitos (2000)
11. Egenhofer, M.: Spherical Topological Relations. Journal on Data Semantics III, 25–49 (2005)
12. Schneider, M., Behr, T.: Topological Relationships between Complex Spatial Objects. ACM Transactions on Database Systems 31, 39–81 (2006)
13. Nedas, K., Egenhofer, M., Wilmsen, D.: Metric Details of Topological Line-Line Relations. International Journal of Geographical Information Science 21, 21–48 (2007)
14. Kurata, Y., Egenhofer, M.: The Head-Body-Tail Intersection for Spatial Relations between Directed Line Segments. In: Raubal, M., Miller, H., Frank, A., Goodchild, M. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 269–286. Springer, Heidelberg (2006)
15. Kurata, Y., Egenhofer, M.: The 9+-Intersection for Topological Relations between a Directed Line Segment and a Region. In: Gottfried, B. (ed.) 1st International Symposium for Behavioral Monitoring and Interpretation, pp. 62–76 (2007)
16. OpenGIS Consortium: OpenGIS Simple Features Specification for SQL (1998)
17. Frank, A.: Qualitative Spatial Reasoning about Distances and Directions in Geographic Space. Journal of Visual Languages and Computing 3, 343–371 (1992)
18. Freksa, C.: Using Orientation Information for Qualitative Spatial Reasoning. In: Frank, A., Campari, I., Formentini, U. (eds.) GIS 1992. LNCS, vol. 639, pp. 162–178. Springer, Heidelberg (1992)
19. Schlieder, C.: Reasoning about Ordering. In: Frank, A., Kuhn, W. (eds.) COSIT 1995. LNCS, vol. 988, pp. 341–349. Springer, Heidelberg (1995)
20. Clementini, E., Billen, R.: Modeling and Computing Ternary Projective Relations between Regions. IEEE Transactions on Knowledge and Data Engineering 18, 799–814 (2006)
21. Billen, R.: Nouvelle Perception De La Spatialité Des Objets Et De Leurs Relations. Développment D'une Modélisation Tridimensionnelle De L'information Spatiale. Department of Geography, Ph.D. Thesis. University of Liège, Liège, Belgium (2002)
22. Billen, R., Zlatanova, S., Mathonet, P., Boniver, F.: The Dimensional Model: A Framework to Distinguish Spatial Relationships. In: Richardson, D., van Oosterom, P. (eds.) Advances in spatial data handling: 10th International Symposium on Spatial Data Handling, pp. 285–298. Springer, Heidelberg (2002)
23. Billen, R., Clementini, E.: Etude Des Caractéristiques Projectives Des Objets Spatiaux Et De Leurs Relations. Revue Internationale de Géomatique 14, 145–165 (2004)

24. Berger, M.: Géométrie 3 : Convexes Et Polytopes, Polyèdres Réguliers, Aires Et Volumes. Cedic/Fernand Nathan, Paris (1978)
25. Alexandroff, P.: Elementary Concepts of Topology. Dover Publications, Mineola (1961)
26. Freksa, C.: Temporal Reasoning Based on Semi-Intervals. Artificial Intelligence 54, 199–227 (1992)
27. Galton, A.: Lines of Sight. In: AI and Cognitive Science 1994, pp. 103–113. Dublin University Press (1994)
28. Egenhofer, M., Mark, D.: Modeling Conceptual Neighborhoods of Topological Line-Region Relations. International Journal of Geographical Information Systems 9, 555–565 (1995)
29. Gottfried, B.: Reasoning about Intervals in Two Dimensions. In: Thissenm, W., Pantic, M., Ludema, M. (eds.) IEEE International Conference on Systems, Man and Cybernetics, pp. 5324–5332 (2004)
30. Van de Weghe, N., De Maeyer, P.: Conceptual Neighbourhood Diagrams for Representing Moving Objects. In: Akoka, J., Liddle, S.W., Song, I.-Y., Bertolotto, M., Comyn-Wattiau, I., van den Heuvel, W.-J., Kolp, M., Trujillo, J., Kop, C., Mayr, H.C. (eds.) ER Workshops 2005. LNCS, vol. 3770, pp. 228–238. Springer, Heidelberg (2005)
31. Moratz, R., Renz, J., Wolter, D.: Qualitative Spatial Reasoning about Line Segments. In: Horn, W. (ed.) 14th European Conference on Artificial Intelligence, pp. 234–238. IOS Press, Amsterdam (2000)
32. Van de Weghe, N., Kuijpers, B., Bogaert, P., De Maeyer, P.: A Qualitative Trajectory Calculus and the Composition of Its Relations. In: Rodriguez, A., Cruz, I., Egenhofer, M., Levashkin, S. (eds.) GeoS 2005. LNCS, vol. 3799, pp. 60–76. Springer, Heidelberg (2005)

Noisy Road Network Matching

Yago Diez[1,⋆], Mario A. Lopez[2], and J. Antoni Sellarès[1,⋆]

[1] Universitat de Girona, Spain
{ydiez,sellares}@ima.udg.edu
[2] University of Denver, USA
mlopez@du.edu

Abstract. Let $\mathcal{N}$ and $\mathcal{M}$ be two road networks represented in vector form and covering rectangular areas R and R', respectively, not necessarily parallel to each other, but with $R' \subset R$. We assume that $\mathcal{N}$ and $\mathcal{M}$ use different coordinate systems at (possibly) different, but known scales. Let $\mathcal{B}$ and $\mathcal{A}$ denote sets of "prominent" road points (e.g., intersections) associated with $\mathcal{N}$ and $\mathcal{M}$, respectively. The positions of road points on both sets may contain a certain amount of "noise" due to errors and the finite precision of measurements. We propose an algorithm for determining approximate matches, in terms of the *bottleneck* distance, between $\mathcal{A}$ and a subset $\mathcal{B}'$ of $\mathcal{B}$. We consider the characteristics of the problem in order to achieve a high degree of efficiency. At the same time, so as not to compromise the usability of the algorithm, we keep the complexity required for the data as low as possible. As the algorithm that guarantees to find a possible match is expensive due to the inherent complexity of the problem, we propose a *lossless filtering* algorithm that yields a collection of candidate regions that contain a subset S of $\mathcal{B}$ such that $\mathcal{A}$ *may* match a subset $\mathcal{B}'$ of S. Then we find possible approximate matchings between $\mathcal{A}$ and subsets of S using the matching algorithm. We have implemented the proposed algorithm and report results that show the efficiency of our approach.

1 Introduction

Spatial analysis of GIS data often requires relating data obtained from two or more different sources. The process of aligning the geometry of two data sets, usually with the purpose of transferring the attributes from one set to the other, is an example of this, referred to as conflation. This type of operation may be needed, for instance, in order to relate a TIGER® file [15] of city streets with a digitized legacy map or with a non-georeferenced map developed in-house. Under such circumstances, it is possible that the two data sets are based on two different coordinate systems, one or both of which are not known precisely or at all.

We consider here the problem of matching a vector data set $\mathcal{M}$ to a subset of another vector data set $\mathcal{N}$, where $\mathcal{N}$ and $\mathcal{M}$ are specified using different

⋆ Partially supported by the Spanish Ministerio de Educación y Ciencia under grant TIN2007-67982-C02-02.

and unknown coordinate systems at different but known scales (map scale is basic information, usually known, even in legacy maps). Here, $\mathcal{N}$ and $\mathcal{M}$ cover rectangular areas R and R', respectively, with $R' \subset R$, and the goal is to find an affine transformation (a sequence of rotations, scalings, translations) that locates R' within R. The data sets may contain small imprecisions in geometry, where each true point lies within a circle of radius ϵ from its specified coordinates. We refer to this problem as **noisy road network matching** and define it formally at the end of this section. Note that even though the algorithms are described in terms of road network data, they can be easily extended to other vector formats such as coverages.

While there is a growing body of literature for the problem of relating two data sets with the same coordinate system (see [11,12,14], for example) or, equivalently, with different but known coordinate systems, [3] is the only work we are aware of capable of matching data originating from two different and partially unknown coordinate systems. In [3], the authors assume that R and R' are parallel to each other, thus the target transformation consists of scaling and translation only, and cannot accommodate rotations. In our work, we assume that the map scales are known (as is usually the case for most maps), but the rectangular areas may be arbitrarily rotated and translated with respect to each other. Thus, our work and that of [3] tackle different but, in a sense, complementary aspects of a similar problem.

An example of the type of data we are dealing with is illustrated in Figure 1.

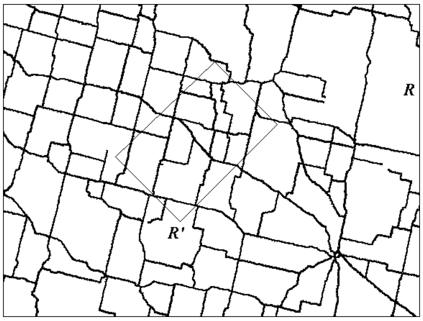


Fig. 1. A road network $\mathcal{N}$ and subnetwork $\mathcal{M}$ with bounding box R'

In this paper we present an algorithm for the case of Road Network Matching using the results of [5] as a starting point. One of the specific characteristics of this problem is the presence of "additional" information such as adjacency relationships between intersections or road types. Our algorithm benefits from this auxiliary information. However, in order to keep the algorithm independent of our data sources, we have only used information that can be considered as "evident" to observers.

Our algorithm first examines the line segments of $\mathcal{N}$ and $\mathcal{M}$ in order to extract the endpoints of the input segments. For each such endpoint p, we store a list

Adj(p) of its neighbors. For road networks, it is reasonable to assume that the degree of every point is bounded by a small constant, representing the maximum number of roads that may meet at any intersection. This process results in sets $\mathcal{A}$ and $\mathcal{B}$ of *road points* from $\mathcal{M}$ and $\mathcal{N}$ respectively, each of which may either be an *intersection point* (a point of degree bigger than two), a *segment interior point* (a point of degree two) or a *segment endpoint* (a point of degree one). Here, we assume that the shape of the map is well described by the positions of intersection points, as this information tends to be stable across maps of the same region. Accordingly, from now on we will only consider intersections whenever we refer to *(road) points.* Also, since the map scales are known, it is easy to represent both point sets using the same normalized scale. Additionally, we also associate with every road point a list of the type of roads that converge on it (such as highway, presence of tunnel, one-way street, etc.). The road categories are defined in terms of the Census Feature class codes (*CFCC*) present in TIGER/line® files.

We assume that the position of points in both sets $\mathcal{A}$ and $\mathcal{B}$ may contain a certain amount of noise due to the measurement errors or the finite precision of input devices. This is modelled by assuming that the actual position of the points involved may vary by a distance up to a fixed quantity ϵ.

We are now ready to formally define our problem.

Let $\mathcal{A}$ and $\mathcal{B}$ be two road point sets of the same cardinality, expressed using the same normalized scale (i.e., a unit of distance means the same thing in both sets).

An *adjacency-degree preserving bijective mapping* $f : \mathcal{A} \to \mathcal{B}$ maps each point $a \in \mathcal{A}$ to a distinct and unique point $f(a) = b \in \mathcal{B}$ so that $|Adj(a)| = |Adj(b)|$.

Let $\mathcal{F}$ be the set of all adjacency-degree preserving bijective mappings between $\mathcal{A}$ and $\mathcal{B}$. The *bottleneck distance* between $\mathcal{A}$ and $\mathcal{B}$ is defined as:

$$d_b(\mathcal{A}, \mathcal{B}) = \min_{f \in \mathcal{F}} \max_{a \in \mathcal{A}} d(a, f(a)) .$$

The **Noisy Road Network Matching (NRNM)** problem can now be formulated as follows. Given two road point sets $\mathcal{A}$, $\mathcal{B}$, $|\mathcal{A}| = n$, $|\mathcal{B}| = m$, $n \leq m$, and $\epsilon \geq 0$, determine a rigid motion (translations plus rotations) τ for which there exists a subset $\mathcal{B}'$ of $\mathcal{B}$, $|\mathcal{B}'| = n$, such that $d_b(\tau(\mathcal{A}), \mathcal{B}') \leq \epsilon$.

If τ is a solution to the **NRNM** problem, every road point of $\tau(\mathcal{A})$ approximately matches a distinct and unique point of $\mathcal{B}'$ of the same degree, and we say that $\mathcal{A}$ and the subset $\mathcal{B}'$ of $\mathcal{B}$ *approximately match* or are *noisy congruent.*

We now discuss the main differences between our paper and the similar problem described in [3]. We then briefly address the results upon which our approach is based.

– In choosing intersections, our goal is to use "prominent" geographical locations, such as main road intersections, landmark places, etc, that are reasonably expected to appear in any map. Under this assumption, it is important that points in $\mathcal{A}$ be mapped uniquely to points in B. In this context, we use the *bottleneck distance* because we believe it is more realistic than the *Hausdorff distance* used in [3]. The reason for this is that Hausdorff distance does not require that points in the two sets to be matched one-to-one. However, for our problem it seems reasonable to require that no two road points in one map be matched to the same road point in the other. This could lead to an input set $\mathcal{A}$ matching a set $\mathcal{B}$ with fewer points, a situation that is undesirable given our goals. Figure 2 illustrates this problem. A possible set $\mathcal{B}$ (indicated by dots) matches a "smaller" set (indicated by squares) under the Hausdorff distance, but not under bottleneck distance.

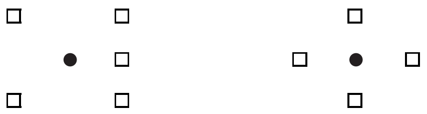


Fig. 2. Two sets of road points that match under Hausdorff distance (for a suitable ϵ) even though they should not match in the *NRNM* problem

- We allow rotations and translations only. Scaling is possible only by assuming that the map scales are known, as is usually the case in practice.
- Another important difference is that the algorithm in [3] computes candidate matchings by finding the transformation T (scaling plus translation) that *exactly* matches a pair of points from $\mathcal{A}$ to a pair from $\mathcal{B}$. It is not hard to find examples where this type of transformation would miss a correct match between $\mathcal{A}$ and a subset of $\mathcal{B}$. For example, Figure 3 contains two sets (one indicated by squares and the other by dots) of three points each. Part (a) of the figure shows that a match is possible (for a suitable ϵ). After computing T using the leftmost two points of each set (illustrated in part (b)), the two leftmost pairs coincide but we have increased the distance for the third pair to a value bigger than ϵ, wrongly concluding that no match can occur. The same happens if we choose any other pair.
- Finally, depending on the values considered for parameter ϵ the practical complexity of the problem changes dramatically. In [3] only very small values of ϵ are considered (the resulting problem is called *Nearly Exact Matching* in the reference). Our algorithm works for any value of ϵ and, in Section 4, we present further discussion of the role of ϵ.

The algorithm presented in this paper uses ideas from Noisy Point Set Matching (**NPSM**) algorithms. The study of this problem was initiated by Alt *et al*

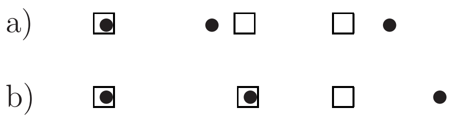


Fig. 3. a) Three pairs of points, where the pair on the left overlaps and the distances between the two points of each other pair is the same and less than ϵ. The two sets match in the position presented. b) If we force two pairs to overlap (by scaling one of the sets), no match occurs.

[1] who presented an exact $O(n^8)$ time algorithm for finding the transformation that produces the best match between two sets of the same cardinality n. Combining [1] with the techniques proposed by Efrat *et al* [6] the time can be reduced to $O(n^7 \log n)$. Both algorithms suffer from high computational cost due to the inherent complexity of the problem and both use complex data structures that are difficult to implement. The running time is improved here by a lossless filtering preprocess similar to the one proposed in [5] for the problem of colored point set matching. For the choice of practical values of parameter ϵ we also use concepts from pattern matching, as presented in [13].

Finally, as this paper tackles a practical problem, we believe that the evaluation of its efficiency must be done with the greatest rigor. With this goal , we follow the ideas presented in [10] in order to make our experiments as meaningful as possible.

2 NRNM Algorithm

In this section we present a nontrivial adaptation of the algorithm in [5] to the **NRNM** problem. We focus on the aspects that are directly relevant to our problem and only briefly sketch the previous algorithm.

The **NRNM** algorithm consists on two phases: enumeration and testing. The enumeration phase makes the problem finite by partitioning all possible rigid motions of $\mathcal{A}$ into equivalence classes and choosing a representative motion τ for each class. The testing phase runs a matching algorithm between every set $\tau(\mathcal{A})$ and set $\mathcal{B}$. To speed up calculations during this phase, additional information about road points is considered. The information used should be evident to any observer and, thus, largely independent from the source of the data. Examples of this type of information include the type of road considered and the connectivity of each road point. This will be further discussed in Section 3.

2.1 Enumeration

Generating every possible rigid motion that brings set $\mathcal{A}$ onto a subset of $\mathcal{B}$ is infeasible due to the continuous nature of movement. Following the algorithm presented in [1] and also used in [6,5], we partition the set of all rigid motions into equivalence classes in order to make their handling possible.

This is achieved by realizing that, for any motion that is a solution of the **NRNM** problem, there exists another that is also a solution with the property that two points in $\mathcal{A}$ lie in the boundaries of the noise disks of the two points in $\mathcal{B}$ to which they are matched.

Consequently, $O(n^2m^2)$ 4-tuples of points are considered. For each 4-tuple, we need to work with $O(nm)$ pairs of points in order not to miss any possible matching, obtaining $O(nm)$ *critical events*. The computation of the critical events requires working with high degree algebraic curves known as *coupler curves* of four bar linkages [9]. Summed over all tuples the total number of critical events encountered in the course of the algorithm is $O(n^3m^3)$. For each of these critical events we will need to run the testing algorithm presented in Section 2.2.

The $O(n^2m^2)$ 4-tuples and $O(n^3m^3)$ critical events express the intrinsic complexity of the problem we are dealing with. They also represent the source of most of the computational effort needed to solve the problem. We will provide experiments to support this statement in Section 4. In general, this cost cannot be decreased as there exist situations where the maximum costs are met. However, in the case of the **NRNM** problems there are ways to reduce it.

Approaches to reduce the cost can be divided into two main types: a) Cutting the computational time needed for every critical event, and b) Reducing the number of 4-tuples and, thus, critical events to be examined. In the first group we find the use of efficient data structures and the "fine tuning" of their implementations. In the second we consider two main strategies. The first one aims at dividing the road points in finer categories and is achieved by considering more information associated to each of them. This information includes adjacency and other evident information, such as *CFCC* codes. These codes are described in the TIGER/lines® technical documentation as "codes that describe the most noticeable characteristics of a feature". More details on this can be found in Section 3.

2.2 Testing

For each 4-tuple we consider every critical event resulting from the enumeration part. These critical events are represented as a series of subintervals of $[0, 2\pi)$. We consider a value ϕ in each of these subintervals and its corresponding motion τ_ϕ. We need to test if there exists a perfect matching between $\tau(\mathcal{A})$ and some subset of $\mathcal{B}$. Whenever the testing part determines a matching of cardinality n we record τ_ϕ and proceed.

We provide Algorithm 1 as a summary of the enumeration and testing sections.

The testing algorithm uses ideas from graph theory to look for perfect matchings between $\tau(\mathcal{A})$ and some subset of $\mathcal{B}$. The algorithm uses two main operations that need to be performed efficiently: a) *neighbor* $(D(\mathcal{T}), q)$: for a query point q, use data structure $D(\mathcal{T})$ which stores a point set $\mathcal{T}$, to return a point in $\mathcal{T}$ whose distance to q is at most ϵ, or $\emptyset$, if no such element exists; b) *delete*$(D(\mathcal{T}), s)$: deletes point s from $D(\mathcal{T})$. For our implementation we use the *skip quadtree*, a data structure that combines the best features of a quadtree and a skip list [7]. This data structure allows us to achieve an amortized time cost of $O(n \log m)$ to test every critical event. This yields a total cost of $O(n^4m^3 \log m)$ for the whole of the algorithm which is a cost analogous to the one obtained in [6] but using simpler data structures. For more details on the computational costs of our algorithm, see [5].

3 Lossless Filtering Preprocess

Most of the computational cost of the **NRNM** algorithm arises from having to consider all possible n^2m^2 quadruples of points described in Section 2.1. Although in some cases it will be necessary to examine all 4-tuples (for example when both sets have the same cardinality), in others there will be subsets of $\mathcal{B}$ that cannot possibly contain matches (for example because they do not contain

Algorithm 1. Search for noisy matching $(\mathcal{A}, \mathcal{B})$

```
{Generate all possible equivalence classes:
ENUMERATION}
for all 4-tuples $a_i, a_j, b_k, b_l$ do
    for every couple $(a_m, b_p)$ do
        Calculate curve $\sigma_{ijklm}$
        $I_{m,p} \leftarrow Intersection((b_p)^\epsilon, \sigma_{ijklm}(x))$
        $\{Critical\_Events\} = \{Critical\_Events\} \cup I_{m,p}$
    end for
end for

{Search for possible matching in every equivalence class: TESTING}
x=0
while $x < 2\Pi$ do
    $x \leftarrow$ next critical event
    $\tau \leftarrow$ associated_rigid_motion($x$)
    if $(matching(\tau(\mathcal{A}), \mathcal{B}))$ then
        {Use algorithm in the "testing" section }
        Record(($\tau$))
    end if
end while
```

enough points) and we will be able to avoid looking at the 4-tuples that contain points in these regions. We adapt the lossless filtering preprocess presented in [5]. This preprocess discards subsets of $\mathcal{B}$ according to a series of geometric parameters that are invariant under rigid motion. These parameters help us to describe and compare the shapes of $\mathcal{A}$ and the different subsets of $\mathcal{B}$ that we explore. This ability to discard parts of $\mathcal{B}$ amounts to a pruning of the search space which results in a reduction of the total computational time.

The lossless filtering preprocess yields a collection of *candidate zones* which are regions that contain a subset $\mathcal{S}$ of $\mathcal{B}$ such that $\mathcal{A}$ may approximately match one or more subsets $\mathcal{B}'$ of $\mathcal{S}$. By doing this we transform the initial problem in a number of smaller and independent subproblems and discard those parts of the search space that do not meet the requirements imposed by the geometric parameters. After this preprocessing, we solve the **NRNM** problem between $\mathcal{A}$ and every $\mathcal{S}$ with the algorithm we have already presented.

Figure 4 describes the structure of our solution.

3.1 Lossless Filtering Algorithm

The lossless filtering preprocess consists of two algorithms: quadtree construction and search. The quadtree construction algorithm also consists of two subparts: A compressed quadtree building algorithm that uses the points in $\mathcal{B}$ as sites (without considering their degree), and an algorithm that adds the information related to the geometric parameters being used at each node. The search algorithm traverses the quadtree looking for candidate zones. Below, we provide a more detailed explanation of our solution.

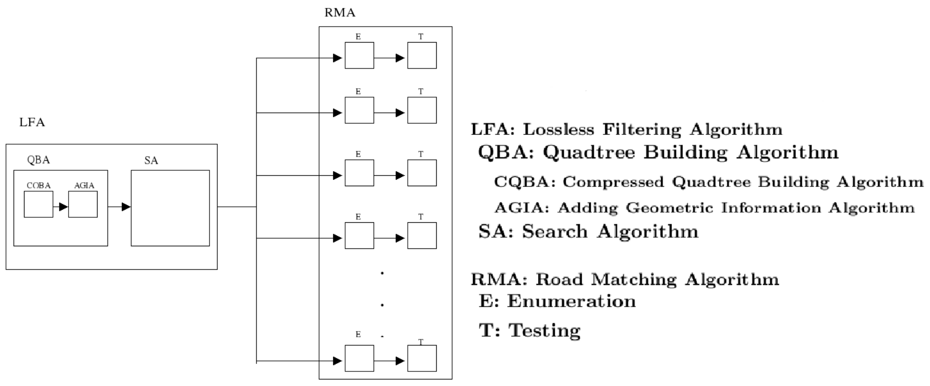

RMA
E
T
LFA
QBA
SA
CQBA
AGIA
LFA: Lossless Filtering Algorithm
QBA: Quadtree Building Algorithm
CQBA: Compressed Quadtree Building Algorithm
AGIA: Adding Geometric Information Algorithm
SA: Search Algorithm
RMA: Road Matching Algorithm
E: Enumeration
T: Testing

Fig. 4. Overview of all the algorithms used

The subdivision of $\mathbb{R}^2$ induced by a certain level of the quadtree consists of axis-parallel squares. At first this does not seem to help because of the requirement to allow set $\mathcal{A}$ to undergo rotations, but we can easily avoid this problem by just searching for a certain axis-parallel square in the quadtree big enough to contain set $\mathcal{A}$ even if it appears rotated. As stated before, we will also require that the square we are looking contains a portion of $\mathcal{B}$ similar to $\mathcal{A}$, in terms of some (rotation invariant) geometric parameters. By doing this, we will be able to temporarily forget about all the possible motions that set $\mathcal{A}$ may undergo and concentrate on those zones of the quadtree where they may actually appear by performing a type of search that is much better suited to the quadtree data structure.

The geometric parameters we use are divided into two main groups:

– **General parameters**. In this first group we find all parameters that can be considered in point set matching problems where categories are considered. In our cases the categories depend on the degree of adjacency of the (road) points:
  a) Parameters based on the fact that we are working with (road) point sets: number of points and histogram of degrees present in the point set.
  b) Parameters based on distances between (road) points: maximum and minimum distance between points of every different degree.
– **Road network specific parameters.** c) Histogram of the CFCC codes present in the point set.

For every geometric parameter we will define a *parameter compatibility criterion* that will allow us to discard zones that cannot possibly contain a subset $\mathcal{B}'$ of $\mathcal{B}$ that approximately matches $\mathcal{A}$. See Figure 5 for an example.

We have chosen these parameters, amongst many others, because they are easy to obtain and fairly independent from the data source. In our experiments we have worked mainly with TIGER/lines® data files. These files contain much information that may have been of great help for our algorithm, such as a unique id number for every road intersection point present in the database. Even when this information would have greatly reduced the time of our calculations, one

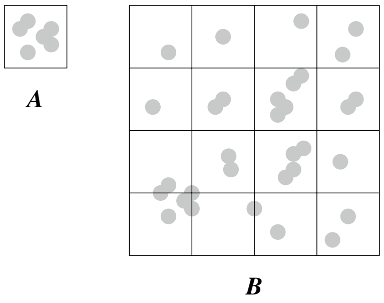


Fig. 5. There cannot be any $\mathcal{B}'$ that approximately matches $\mathcal{A}$ within the four top-left squares of B because $\mathcal{A}$ contains six disks (representing points) and the squares contain only five

of our goals was to ensure the usability of our algorithm in different situations, so we have restricted our attention to parameters "evident to any observer". With this in mind, we have used the degree of every road point and also the CFCC codes, as they describe the most noticeable characteristics of a feature. Furthermore, both degree and CFCC codes must be present in all records of a TIGER/lines® file.

The time cost of adding these geometric parameters to the quadtree of set $\mathcal{B}$ is in $O(m^2)$.

It is important to stress that ours is a conservative algorithm, so we do not so much look for candidate zones as rule out those regions where no candidate zones may appear. A technical issue that arises at this point is that, although

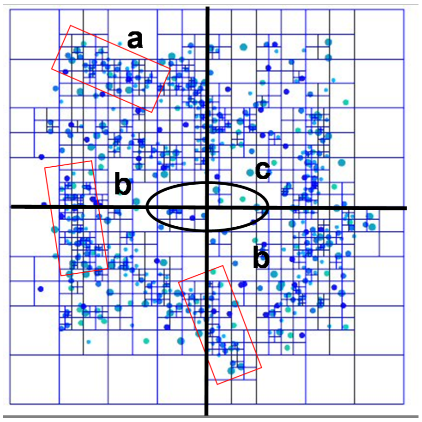


Fig. 6. Position of the candidate zones in the grid. Overlapping: (a) a single grid-square (corresponding to a single quadtree node), (b) two (vertically or horizontally) neighboring nodes, or (c) four neighboring nodes. In this example we observe occurrences of set $\mathcal{A}$ in zones of the first two types and an ellipse showing where occurrences of the third type (not present) would appear.

our intention was to describe our candidate zones as squares of size s, this will not always be possible, and we will also have to consider groups of two or four squares of size s. Notice that the earlier the discards are made, the bigger the subsets of $\mathcal{B}$ that are discarded.

Search Algorithm

The search algorithm identifies all squares that cover a subset of $\mathcal{B}$ where candidate zones, compatible with $\mathcal{A}$, may be located. This results in three different kinds of candidate zones associated to, respectively one, two or four nodes (see Figure 6) of the tree. The subsets $\mathcal{B}'$ that we are looking for may lie anywhere inside those zones.

Consequently, the zones considered throughout the search are easily described in terms of the nodes of $\mathcal{Q}_{\mathcal{B}}$ and gradually decrease in size, until they reach s, following the descent of the quadtree. Given that two or four nodes defining a candidate zone need not be in the same branch of $\mathcal{Q}_{\mathcal{B}}$, at some point we may need to explore two or four branches simultaneously. Algorithm 2 outlines the main search function.

The Search algorithm runs in $O(m)$ time. The total cost of the lossless filtering algorithm is $O(m^2)$.

3.2 Overall Computational Costs

Combining the costs of the lossless filtering preprocess and the Enumerating and testing algorithms it can be seen that the total cost of the matching algorithm is $O(n^4 m^3 \log m)$. This bound is tight. This shows that from a formal point of view, our process takes, at its worst, the same computational time as the algorithm that does not use the lossless filtering step. Consequently we benefit from any reduction of the computational time that the filtering achieves without any increase in the asymptotic costs.

The cost is high mainly because of the inherent geometric complexity of the problem. In Section 4 we quantify the important improvement in computational costs achieved by using the lossless filtering algorithm, and also by discarding the maximum number of 4-tuples, as described in Section 2.2.

4 Implementation and Results

In this section we present experiments that show the degree of efficiency of our algorithms. We have worked with TIGER/lines® files for the counties of Denver (files TGR08031), Adams (TGR08001) and Arapahoe (TGR08005). As we have only used information concerning the position of road points, their connectivity and their associated CFCC codes, it was enough to work with .RT1 files. These files can be found at [15] and the corresponding documentation at [16].

We have implemented all our algorithms in C++ under a Linux environment. We used the g++ compiler without compiler optimizations. All tests were run on a Pentium D machine with a 3 GHz processor.

Algorithm 2. Search_1(node N)

```
for all children $S$ of $N$ do
  if ($S$ is parameter compatible with $\mathcal{A}$ ) then
    if ( We have not reached the node size to stop the search) then
      Call Search_1($S$)
    else {We have found a candidate node}
      Report candidate zone
    end if
  end if
end for
{Continue in pairs of nodes if necessary (four possibilities)}
for all $S_1, S_2$ pairs of neighboring children of $N$ do
  if (The couple $(S_1, S_2)$ is parameter compatible with $\mathcal{A}$) then
    if ( We have not reached the node size to stop the search) then
      Call Search_2($S_1, S_2$)
    else {We have found a candidate pair}
      Report candidate zone
    end if
  end if
end for
{Finally, continue in the quartet formed by the four children if necessary}
$(S_1, S_2, S_3, S_4)$: Quartet formed by the children of $N$.
if ($(S_1, S_2, S_3, S_4)$ are parameter compatible with $\mathcal{A}$ ) then
  if ( We have not reached the node size to stop the search) then
    Call Search_4 ($S_1, S_2, S_3, S_4$)
  else {We have found a candidate quartet}
    Report candidate zone
  end if
end if
```

We consider the set obtained from this data as our set $\mathcal{B}$. To obtain set $\mathcal{A}$, we choose a subset of set $\mathcal{B}$. To choose this subset we randomly choose a point p in $\mathcal{B}$, consider an axis-parallel rectangle that has p lower-left vertex. Then we rotate it a random angle obtaining a set as depicted in Figure 1. Once we have chosen our set $\mathcal{A}$, we apply a random transformation to it that includes:

– Random rotation around one of its points.
– A translation of random vector whose components are bounded by the dimensions of set $\mathcal{B}$.
– A small perturbation applied independently to each of its points in order to simulate noise in data. The modulus of this perturbation is bounded by ϵ.

Unless stated otherwise, set $\mathcal{B}$ is fixed and sets $\mathcal{A}$ of varying sizes are chosen as described above.

The amount of noise ϵ allowed in the data impacts the practical complexity of the problem and, consequently, the computational times obtained. We illustrate this statement with the two extreme cases.

1. $\boldsymbol{\epsilon = 0}$. In this case, the problem would most likely, in practice, not have a solution. As the calculations made to build set $\mathcal{A}$ introduce a certain amount of error (due to rounding, truncation and changes over time in the underlying data) if we required ϵ to be exactly 0, then there would be no exact match to be found. This behavior can also apply to very small values of ϵ.
2. $\boldsymbol{\epsilon = \infty}$. In this case we consider a value of ϵ too big (meaning, at least bigger than the diameter of set $\mathcal{B}$). In this case, any rigid motion that brings set $\mathcal{A}$ inside a circle of radius ϵ that contains $\mathcal{B}$ would be a solution to our problem. This behavior is clearly undesirable. Such a big value of ϵ stands for knowing very little about the real positions of the points in set $\mathcal{B}$ and any solution found based on this type of data is meaningless.

These two extreme cases show that too small values of ϵ may result in the algorithm missing the solution due to numerical problems, and too big ones may result in meaningless solutions. It is thus very important to consider *reasonable* values of ϵ. To do this, we followed the ideas presented in [13] and studied the empirical values of the *average shortest distance* of set $\mathcal{B}$, $t_{\mathcal{B}}$ defined in the reference as $t_{\mathcal{B}} = \frac{r_{\mathcal{B}}}{2|\mathcal{B}|}$ where $r_{\mathcal{B}}$ is the radius of set $\mathcal{B}$. We have tested different values of ϵ and present results on $\epsilon = t_{\mathcal{B}}$. We also provide some further discussion on this subject on Section 4.4.

4.1 Additional Improvements

In addition to the strategies described until now to reduce the running times of the algorithms (that do so from a theoretical point of view) we have also used some practical strategies that we describe now.

- We observed that the number of road points of degree greater than 5 in our data sets was much smaller than the number of points of degree less than or equal to 5. Taking advantage of this, in every candidate zone we performed an intermediate filtering step looking for a match but considering only road points of degree greater than 5. If this matching was found, we proceeded with the rest of the points and otherwise there was no need to keep on searching in that zone.
- When working with a particular 4-tuple of points, all remaining points in $\mathcal{A}$ and $\mathcal{S}$ have to be tested to decide if they may be matched. This decision is affirmative if one of the points lies in the interior of a coupler curve described by the other. This involves complex calculations, but points that are "too far away" from, for example, the point that we take as the initial point of the coupler curve will never be matched. To reduce the number of pairs considered, we approximated the diameter of the coupler curves by sampling some points on them and calculating their discrete diameter. This resulted in avoiding calculations for most of the points, improving the techniques presented in [5].

Finally, we have to consider the fact that, as we are only looking for one possible match, the computational costs of our experiments also depend on how "early" the algorithm visits the 4-tuple that leads to the solution. To minimize the effect of this, in those discussions where the time spent by the algorithm is the main subject (as is the case of Section 4.2) we will present average times (spent to find a number of sets $\mathcal{A}$ of the same size, chosen randomly) instead of results of just one experiment.

4.2 Effects of the Lossless Filtering Algorithm

The performance of the algorithm depends on the effectiveness of the lossless filtering step and the parameters chosen but, in the worst case, it meets the best (theoretical) running time up to date.

Focusing in our observations of our experiments we can say that in the best case, the initial problem is transformed into a series of subproblems with cardinality n' close to $n = |\mathcal{A}|$, producing a great saving of computational effort. To be fair, we also have to mention that in the worst case, when both sets have similar cardinality, filtering doesn't help. In fact, using the lossless filtering preprocessing step may even increase the running time. In this section we try to quantify this saving in computational time due to filtering. Figure 7 compares the behavior of the matching algorithm with and without the lossless filtering algorithm (represented by times T1 and T2 in the figure, both in seconds). We fixed set $\mathcal{B}$ to be a subset of 5000 road intersections of the county of Denver and tried sets $\mathcal{A}$ of different sizes. For every size, we present the mean of the times spent by ten sets of the same size. This lossless filtering algorithm uses all the parameters described earlier.

We must state that the sizes considered here are small given the huge computational costs of the algorithm without lossless filtering. It is clear from the figure that, even when the theoretical computational costs are still high due to the complexity inherent to the problem, using the lossless filtering results in significant computational savings.

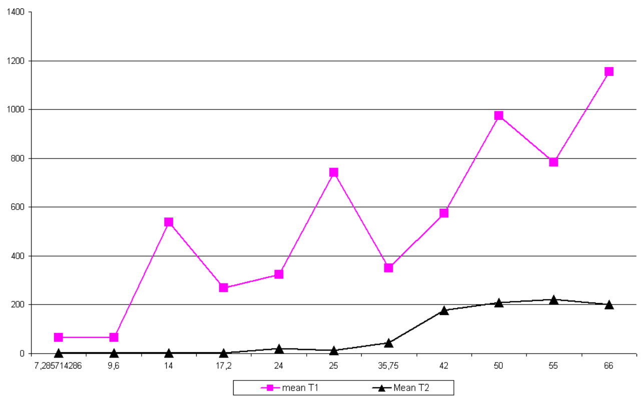


Fig. 7. Running times not using lossless filtering step (T1) and using it (T2). X axis represents the mean values of the cardinals of sets $\mathcal{A}$ in every test and axis Y represents time in seconds.

4.3 Discussion on Geometric Parameters

In this section we provide results that measure the effectiveness of the different geometric parameters used during the lossless filtering algorithm. Table 1 presents the number of candidate zones and computational costs for the search algorithm resulting from: 1) using only the "number of points" (Num.) parameter; 2) using (1) plus the histogram of points degrees (Histo.); 3) using (1), (2) plus the "maximum and minimum distance between points of the same degree" (Dist.) parameters; 4) adding to the previous three the CFCC codes associated to each road intersection point.

For the tests we used two main data sets. The first one contains 12052 road intersections in the county of Denver. The second contains, in addition, data about road intersections in the counties of Adams and Arapahoe totaling 39950 road intersections. We run the lossless filtering algorithm for different sets $\mathcal{A}$ of varying sizes. Table 1 presents the number of candidate zones obtained for every filtering criteria and the time consumed in each case by the search algorithm, the time needed to build the quadtree data structure was approximately 16 seconds for the first data set, and approximately 150 seconds for the second.

Table 1. Effects of the different geometric parameters on the lossless filtering algorithm

$\|\mathcal{A}\|$	$\|\mathcal{B}\|$	Num.		Num./Histo.		Num./Histo./Dist.		Num./Histo./Dist./CFCC	
		#zones	Time(s)	*#zones*	Time(s)	*#zones*	Time(s)	*#zones*	Time(s)
25	12052	217	$\ll 0.01$	188	$\ll 0.01$	187	$\ll 0.01$	31	$\ll 0.01$
100	12052	194	$\ll 0.01$	61	$\ll 0.01$	52	0.19	52	0.19
250	12052	57	$\ll 0.01$	19	$\ll 0.01$	19	0.16	9	0.17
500	12052	21	$\ll 0.01$	17	$\ll 0.01$	16	$\ll 0.01$	10	$\ll 0.01$
750	12052	20	$\ll 0.01$	15	$\ll 0.01$	14	5.41	8	5.41
1000	12052	19	$\ll 0.01$	10	$\ll 0.01$	8	1.57	6	1.59
2000	39925	52	$\ll 0.01$	16	$\ll 0.01$	12	7.55	6	7.53
5000	39925	8	$\ll 0.01$	4	$\ll 0.01$	4	0.12	2	0.14
10000	39925	12	$\ll 0.01$	4	$\ll 0.01$	4	0.01	2	0.03
12500	39925	11	$\ll 0.01$	4	$\ll 0.01$	4	$\ll 0.01$	2	$\ll 0.01$

We observe that the number of candidate zones is always lower when we use more "elaborate" geometric parameters. The main reduction in the number of candidate zones is usually obtained when the histogram of road intersection degrees is considered, but in most cases this reduction continues when distances between road intersections of the same degree and CFCC codes are considered. We also want to highlight two special cases:

- We noticed that the number of candidate zones obtained for sets of parameters (2) and (3) differ more in the second data set. As this stands for considering the distances of road intersections of the same degree or only their histogram, this may be caused by a more regular distribution of road intersections inside the city of Denver than in the other counties considered.
- In some cases, the number of candidate zones obtained for sets of parameters (3) and (4) is the same. The difference in this case is whether we consider

or not CFCC codes. In some of the test cases, all the roads that formed the intersections of set $\mathcal{A}$ had the same code (A41 which, according to the TIGER/lines® documentation, stands for *Local, neighborhood, and rural road, city street, unseparated*). We studied the distribution of road CFCC codes for both data sets and found that more than 93.5% of the roads in Denver and about 90% on Denver, Adams and Arapahoe fall in this category.

With respect to computational times, the time needed to perform the search algorithm is bigger when we use more geometric parameters, but still much less than the cost of the matching algorithm. In conclusion, the use of more geometric parameters results in the output of less, and better, candidate zones. Moreover, although the use of more geometric parameters slightly increases computational time, the cost of the lossless filtering algorithm is still much smaller than that of the matching algorithm.

4.4 Computational Performance

Here we seek to evaluate the performance of the whole algorithm, with all improvements, in some "real life" situation. We ran tests where set $\mathcal{B}$ corresponds to the road intersection points in the county of Denver (cardinality 12052) and used different sets $\mathcal{A}$ chosen randomly inside rectangles of varying sizes. Table 2 shows:

- The cardinalities of the sets involved in the test.
- The average number of candidate zones examined before finding the solution. This value is presented in order to provide an idea on how effective the filtering step is in transforming the initial problem into a series of smaller subproblems with similar cardinalities between $\mathcal{B}'$ and $\mathcal{A}$.
- Finally, we present data on the number of 4-tuples and pairs examined. This data helps us to highlight where the main computational effort is, as it is directly related to the time spent by the algorithm.

Concerning the mean value of n', denoted $\overline{n'}$, we observe that the filtering algorithm manages to take away most of the complexity of the problem. This works better for smaller sets, as the algorithm is able to locate candidate zones

Table 2. Performance of the complete algorithm

$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{B}\rvert$	$\overline{n'}$	#4-tuples	#pairs	Time(s)
10	12052	55	6	93	15.31
50	12052	111	7	218	31.84
75	12052	481	18	1381	73.17
100	12052	653	1	670	27.38
150	12052	1880	4	3484	243.97
200	12052	1089	5	7918	849.65
250	12052	1088	20	19856	2464.71
300	12052	1888	5	2697	404.56

built by smaller nodes in the quadtree. We can also observe that the total computational cost of the algorithm is greatly influenced by the number of 4-tuples or pairs examined, as tests for sets of similar sizes may need very different computational effort (this is illustrated by the rows corresponding to $|\mathcal{A}| = 200$ and $|\mathcal{A}| = 250$). Consequently, the main part of the computational effort can be attributed to the complexity inherent to the problem. Finally the computational time increases with $|\mathcal{A}|$, although we believe that it is kept to reasonable levels given the complexity of the calculations involved.

Other values of ϵ

To complement the discussion on the values of ϵ provided at the beginning of this section, we present some tests for different values of ϵ.

Table 3. Compared performance for different values of ϵ

ϵ	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{B}\rvert$	$\overline{n'}$	#4-tuples	#couples	Time(s)
$t_{\mathcal{B}}$	10	12052	55	4	25	22.51
$\frac{t_{\mathcal{B}}}{2}$	10	12052	55	1	18	22.48
$t_{\mathcal{B}}$	31	12052	99	162	2892	79.87
$\frac{t_{\mathcal{B}}}{2}$	31	12052	99	1	135	19.23
$t_{\mathcal{B}}$	230	12052	540	13	9192	472.08
$\frac{t_{\mathcal{B}}}{2}$	230	12052	1089	14	22183	2114.26

As expected, as the value of ϵ decreases, so does the number of 4-tuples that lead to a solution (arriving to zero in the limit case described previously). This results in an increase of the total computational time due to having to "search more" before finding a solution. Also, smaller values of ϵ result in fewer 4-tuples and pairs to be checked, or sometimes allow for better filtering and smaller candidate zones. Both of these considerations result in a decrease of computational time. Which of the two tendencies is more important? Basically it depends on the data set being searched, for most of our study cases, smaller values of ϵ required higher computational time, but in some others (generally for bigger realizations of set $\mathcal{A}$) the opposite held. Table 3 shows examples of the two cases and even one where both tendencies produce similar computational times for both values of ϵ.

5 Conclusions

In this paper we have presented, to the best of our knowledge, the first formalization of the **NRNM** problem in terms of the *bottleneck* distance. We have presented theoretical and practical discussions on this algorithm and obtained running times that are fast, in light of the inherent complexity of the problem. Experiments show how using the lossless filtering algorithm helps reduce the running time. This reduction is bigger the more elaborate geometric parameters are considered. We have, however, only used information that should be evident to all observers. Finally, we have also provided some examples on how the degree of noise in data (ϵ) influences the performance of the algorithm. As part of our future work, we intend to study other values of ϵ, such as those that arise directly from the precision os measuring devices, and their relationship to the efficiency of the proposed algorithms.

References

1. Alt, H., Mehlhorn, K., Wagener, H., Welzl, E.: Congruence, similarity and symmetries of geometric objects. Discrete & Computational Geometry 3, 237–256 (1988)
2. Brent, R.P.: Algorithms for Minimization Without Derivatives. Prentice-Hall, Englewood Cliffs (1973)
3. Chen, C., Shahabi, C., Kolahdouzan, M., Knoblock, C.A.: Automatically and Efficiently Matching Road Networks with Spatial Attributes in Unknown Geometry Systems. In: 3rd Workshop on Spatio-Temporal Database Management (STDBM 2006) (September 2006)
4. Chen, C.: Automatically and Accurately Conflating Road Vector Data, Street Maps and Orthoimagery. PhD Thesis (2005)
5. Diez, Y., Sellarès, J.A.: Efficient Colored Point Set Matching Under Noise. In: Gervasi, O., Gavrilova, M.L. (eds.) ICCSA 2007, Part I. LNCS, vol. 4705, pp. 26–40. Springer, Heidelberg (2007)
6. Efrat, A., Itai, A., Katz, M.J.: Geometry helps in Bottleneck Matching and Related Problems. Algorithmica 31, 1–28 (2001)
7. Eppstein, D., Goodrich, M.T., Sun, J.Z.: The skip quadtree: a simple dynamic data structure for multidimensional data. In: 21st ACM Symposium on Computational Geometry, pp. 296–305 (2005)
8. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. SIAM Journal on Computing 2(4), 225–231 (1973)
9. Hunt, K.H.: Kinematic Geometry of Mechanisms, chs. 4,7. Oxford University Press, Oxford (1978)
10. Johnson, D.S.: A Theoretician's Guide to the Experimental Analysis of Algorithms. In: Goldwasser, M., Johnson, D.S., McGeoch, C.C. (eds.) Proceedings of the fifth and sixth DIMACS implementation challenges. American Mathematical Society, Providence (2002)
11. Saalfeld, A.: Conflation: Automated Map Compilation. International Journal on Geographical Information Systems 2(3), 217–228 (1988)
12. Walter, V., Fritsch, D.: Matching Spatial Data Sets: a Statistical Approach. International Journal of Geographical Information Science 13(5), 445–473 (1999)
13. Van Wamelen, P.B., Li, Z., Iyengar, S.S.: A fast expected time algorithm for the 2-D point pattern matching problem. Pattern Recognition 37(8), 1699–1711 (2004)
14. Ware, J.M., Jones, C.B.: Matching and Aligning Features in Overlayed Coverages. In: 6th ACM International Symposium on Advances in Geographic Information Systems, pp. 28–33 (1998)
15. U.S. Census Bureau Topologically Integrated Geographic Encoding and Referencing system, TIGER®, TIGER/Line® and TIGER®-Related Products, `http://www.census.gov/geo/www/tiger/`
16. U.S. Census Bureau TIGER/Line® files, Technical documentation. 1edn. (2006), `http://www.census.gov/geo/www/tiger/tiger2006fe/TGR06FE.pdf`

Detecting Topological Change Using a Wireless Sensor Network

Christopher Farah, Cheng Zhong, Michael Worboys, and Silvia Nittel

Department of Spatial Information Science and Engineering,
University of Maine, Orono, ME, 04469, USA
{cfarah,czhong,worboys,nittel}@spatial.maine.edu

Abstract. Dynamic geographic phenomena, such as forest fires and oil spills, can have dire environmental, sociopolitical, and economic consequences. Mitigating, if not preventing such events requires the use of advanced spatio-temporal information systems. One such system that has gained widespread interest is the *wireless sensor network* (WSN), a deployment of *sensor nodes* – tiny untethered computing devices, which run on batteries and are equipped with one or more commercial off-the-shelf or custom-made sensors and a radio transceiver. This research deals with initial attempts to detect topological changes to geographic phenomena by an environmentally deployed wireless sensor network (WSN). After providing the mathematical and technical preliminaries, we define topological change and present in-network algorithms to detect such changes and also, to manage the WSN's resources efficiently. The algorithms are compared against a resource-heavy continuous monitoring approach via simulation. The results show that two topological changes, hole loss and hole formation, can be correctly detected in-network and that energy is greatly saved by our event-driven approach. In future work, we hope to test the algorithms over a broader range of topological changes and to relax some of the network assumptions.

Keywords: Wireless sensor networks, distributed, algorithm, topological change, areal object.

1 Introduction

Dynamic geographic phenomena, such as forest fires and oil spills, can have dire environmental, sociopolitical, and economic consequences. Mitigating, if not preventing such events requires the use of advanced spatio-temporal information systems. One such system that has gained widespread interest is the *wireless sensor network* (WSN), a deployment of *sensor nodes* – tiny untethered computing devices, which run on batteries and are equipped with one or more commercial off-the-shelf or custom-made sensors and a radio transceiver. Beyond catastrophe management, it is expected that WSNs will play a key role in ubiquitous spatial computing, offering researchers and application domain specialists unprecedented opportunities in environmental sensing, monitoring, and analysis. In contrast to other sensing technologies, e.g. LIDAR, a WSN is not restricted to particular types of phenomena: a WSN can detect any *measurand* – a physical parameter, such as light intensity, temperature, or ozone – so long as the corresponding sensor has been developed, and the device can withstand the deployment region's environmental conditions. Given a fixed deployment of sensor nodes over a geographic region, one of two monitoring paradigms can be adopted: *continuous* monitoring or *event-driven* monitoring. In the former, every sensor node in the network samples the environment at a constant rate. In the latter, a sensor node's level of activity is affected by local changes in the measurand. While continuous monitoring offers the best responsiveness possible, it does so at the cost of network resources, i.e. power consumption, and temporal resolution of the environmental data. In the case of event-driven monitoring, regions of low sensor activity demand less of the network's processing, allowing resources to be dedicated to those regions of high activity. By shutting down nodes, however, resolution and responsiveness can decrease. Thus, either approach leads to a compromise in monitoring.

In this paper, dynamic *topological events* with regard to continuous spatial phenomena, events such as hole formation and merging, govern the activity of the network. The motivation is three-fold: (1) to provide a framework for the classification of environmental phenomena by topological behavior, (2) to reduce the cost of data transmission and in-network information processing, and (3) to optimize the number of active nodes through a tiered network. To achieve these goals, nodes are assigned to *clusters* - geographic subsets that partition the region of deployment. For each cluster, one node is promoted to the rank of *cluster head*. By virtue of this hierarchy, updates can be coordinated via cluster heads, allowing the network to "keep up" with global topological changes.

In summary, the contributions of this paper include the following:

- Algorithms for topological event detection in WSNs.
- Algorithms for dynamic resource management.
- Evaluation of proposed algorithms in comparison with the continuous monitoring approach. Events are detected correctly and network energy is saved.

The remainder of the paper is organized as follows. Section 2 provides mathematical and technical preliminaries and summarizes related work. Section 3 describes our approach for event detection in dynamic sensor networks. In section 4, the algorithms for network management and event detection are described in detail. Section 5 outlines the simulation approach to test our algorithms and discusses the corresponding results. Finally, we conclude this paper and discuss future work.

2 Mathematical Preliminaries and Related Work

2.1 Mathematical Preliminaries

A *scalar field* is a spatial domain R, such that for each point $p \in R$, there is a unique scalar value, s_p, assigned to p. In this work, it is assumed that R is planar and that a sensor node associated with point p detects s_p with complete precision. For example, in the case of forest fire monitoring, each point in the region of network deployment resides in a scalar field where temperature is the scalar value. In order to derive topological events from the scalar field, a threshold value is designated relative to the sensor readings, invoking a Boolean response for each point in the spatial domain. Thus, the scalar field is discretized; rather than a continuous range of scalar values, there are precisely two values, 0 and 1. This discretized field is approximated by the network; the resolution depending upon the density of nodes in the WSN, and the proportion that are active. Figure 1 shows the progression from a scalar field to its discretized approximation. This particular example is derived from aerial images of an oil spill off the coast of Spain.

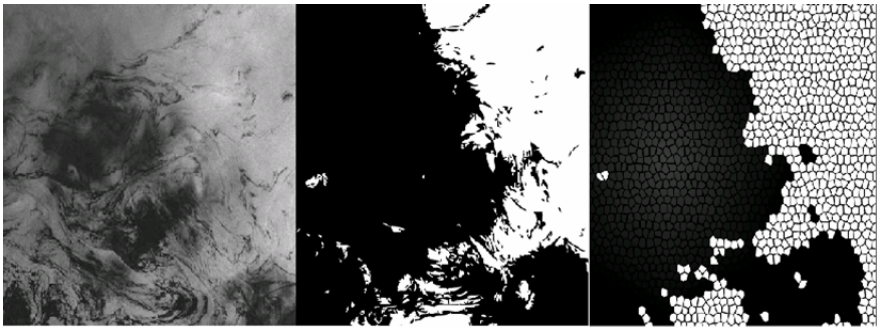


Fig. 1. From scalar field to the discretized, WSN approximation

Let us consider that part of the domain whose corresponding scalar values are 1. This is an areal object [1] and consists of one or more *connected components* (components for simplicity) of R, i.e. regions in which any two points in one such region can be joined by a path, completely contained in the region. Each component has topological properties that can be determined. For the purpose of this investigation, the properties of greatest interest are connectedness and *genus*, which is a count of the number of holes in a component. The motivation is simple: by keeping track of these two properties over consecutive sensor samples, six atomic topological changes can be identified. A *topological change* is a change to an areal object such that there is no homeomorphism between the areal object in its initial state and its final state. Thus, a topological change occurs if an areal object, by virtue of the discretized scalar field's evolution over time, changes status with regard to one or more topological properties. In this paper, "topological change" and "event" are synonymous. The atomic changes to be detected are:

- Hole formation / hole loss
- Self-split / self-merge
- Split / merge

As can be seen in Figure 2, the genus has changed for the cases: hole formation, hole loss, self-split, and self-merge. For the final two topological changes, split and merge, the property connected has changed but not the genus. Jiang and Worboys [1] have proved that any topological change resulting from changes only in genus and connectedness can be expressed as a composition of those above. Therefore, it will suffice to deal with atomic topological changes.

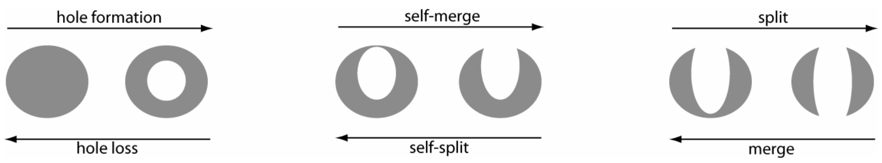


Fig. 2. The six primary topological changes

An *incremental change* to a WSN is the change of sensor status, relative to threshold, of a single node over the entire network at a time t. Such a change will result in zero or more topological changes. In order to capture the changes outlined above, *the neighborhood ring* is introduced. The neighborhood ring is a cyclic data structure stored at a node that maintains its nearest neighbor readings in counterclockwise order. In order to identify its neighbors, a node broadcasts a message at some user-defined fraction of the communication range of the node. Neighbors within this range will have a corresponding entry in their neighborhood ring. Let u be a node and v_1, v_2, …, v_k be the k one-hop neighbors of u within a predefined distance, sequenced in counterclockwise cyclic order around u, where a starting node v_1 is randomly assigned in advance. The neighborhood ring associated with u is a ring data structure $[s_1, s_2, \ldots, s_k]$, starting from v_1 where s_i is the sensor reading of v_i, which are mapped to the Boolean values 0 and 1 as previously described. Figure 3 shows a node u and its neighbors, the underlying discretized scalar field in gray, and two equivalent neighborhood rings. A neighborhood ring - not unlike the connected components of the spatial domain – can be thought of as a collection of contiguous subsequences, i.e., sequences of the form $[0,\ldots\underline{1,1,\ldots,1},\ldots,0]$ or $[\underline{1,\ldots 1},0,0,\ldots 0,\underline{1,\ldots,1}]$, since the neighborhood ring is cyclic, i.e. it can start at any neighboring node. Such a contiguous subsequence is called a *neighborhood component*, and indicates a region of similar sensing. Two neighborhood components are indicated in Figure 3 by dashed rectangles and each can be identified as a sequence of 1s in each neighborhood ring. In order to determine which topological change has occurred, a node that has changed status will initiate a series of tests based upon the neighborhood components of its neighborhood ring.

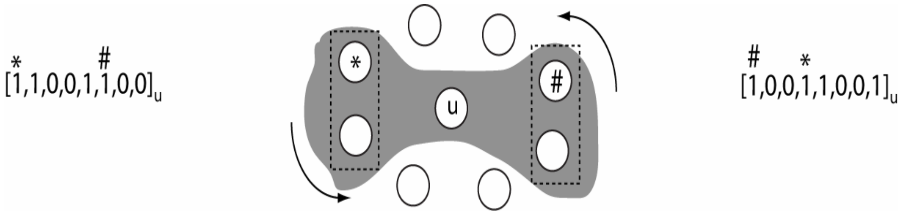


Fig. 3. Two equivalent neighbor rings of node u

2.2 Network Assumptions

A *distributed* computing environment is one in which multiple computing devices or nodes (often) operate in parallel and achieve a common goal by processing available data (when appropriate) in a cooperative fashion, thereby passing the intermediate result as a message to other nodes. Two interesting properties of a distributed WSN are: (i) there is no global clock, and (ii) only local data is stored at a node. The consequences of (i) are that: classic synchronization can not be used to order computations and many events will be temporally incomparable, i.e. it will not be possible for any node to correctly determine the order of topological events, even if the events are temporally ordered in the environment. The consequence of (ii) is that no node has a global view of the network. This does not mean global properties cannot be computed, but that no node has all of the raw data necessary to execute such a computation. While explicit reference to these properties will not be made, it is interesting to bear them in mind in the development that follows.

In addition to the generic distributed system assumptions, we make five additional assumptions related to the operation of the WSN in this research. They are:

1) Neighboring nodes of the same status belong to the same component.
2) Non-neighboring nodes of the same status belong to the same component only if there is a node path between them, such that each node on the path has the same status.
3) Each node knows its own location through either a GPS device or some GPS-less techniques [2, 3], as well as the angle-of-arrival of packets.
4) A node stores its previous and current sensor reading relative to threshold, its neighbor ring, as well as the ID.
5) A node that has been promoted to cluster head stores all data noted above, as well as the genus, the node ID of each adjacent cluster head, and any temporary data structures (See Section 3.1.) needed to complete computations.

The first two assumptions allow sensor data to infer properties of the underlying scalar field. It is easily proved that assumption (2) follows from assumption (1) but it is stated on its own for clarity. Assumption (3) ensures that each node can identify the cluster it belongs to and that a well-ordered neighbor ring can be constructed. Assumptions (4) and (5) ensure that correct topological event detection can be carried out. In particular, if a split or merge occurs, region IDs need to be updated at each node, while a genus update is required for the remaining four topological changes.

2.3 Related Work

2.3.1 Topology Discovery

Current research in hole detection using WSNs has primarily focused on the characterization of the communication graph within the network, i.e. a graph whose vertices are sensor nodes and edges are communication links between sensor nodes. In particular, research has been conducted in order to ensure the effective routing of messages, even in the presence of holes in the communication graph. In [4], the author presents a solution to the problem which requires more computational power than is typically assumed of sensor nodes, and therefore could not be implemented in a distributed setting. Clearly, for long-term unsupervised deployments, a more robust, decentralized solution is needed. In [5], hole detection is distributed and decentralized. However, a ‘flat’ routing approach is applied in order to define and update each hole’s boundary. In other words, data is simply passed via nearest neighbors. While this may be feasible in maintaining the topology of a slowly changing communication graph, the topology of the underlying scalar field is likely to evolve much more rapidly. Additionally, there are different algorithmic expectations: holes in the communication graph result in node failure and therefore, a loss of spatial resolution, while holes in spatial phenomena only result in a change in sensor value, but not the failure of the sensor node itself. Therefore, more efficient and appropriate update techniques are proposed in this work.

The prerequisite for topological change detection is boundary detection in the network. Chintalapudi and Govindan [6] discuss three boundary detection methods and return sufficient data so that the base station can construct an accurate boundary. They do not however, transmit boundary information back into the network, preventing the possibility of localized, in-network updates of the boundary shape and its location. Ding et al. [7] propose localized fault-tolerant boundary and faulty sensor detection using spatial data mining techniques. These techniques have to report all boundary node information to the base station: the topological changes can only be deduced at the base station, after it generates different field snapshots using received data. As in [1], such techniques are not tractable in the case of remote deployments. In this paper, region boundary detection is based on neighboring node value differences: a boundary exists between two nodes if their sensors detect different measurand concentrations, relative to the designated threshold. Based on the detected boundary, all changes are deduced in the network and then are reported back to the base station, on user demand.

2.3.2 Resource Management

In WSNs, particularly for long-term deployments, energy conservation is critical. If communication is not dealt with carefully, then network resources can be unnecessarily expended, which is the largest energy consumer in a sensor network. For example, even if a node broadcasts data intended for a few selected neighbors, every node within transmission range and operating on the same channel must receive and process each packet, whether the packet has computational importance or not. Chen et al. [8] show that the energy consumption ratio, idle:receive:transmit is 1: 2: 2.5. This observation motivates approaches that either reduce the number of active nodes or reduce node contention. Xu et al. [9] develop a geographic adaptive fidelity (GAF) algorithm, which can be implemented in conjunction with any ad-hoc routing algorithm. GAF identifies "equivalent nodes" from a routing perspective: two nodes are equivalent if the cost of routing messages through one is the same as the other. Thus, the network is partitioned into virtual grids. Within each grid, most nodes can be set to sleep, so long as they are not a source, sink or critical intermediate node within the routing chain. Simulation results demonstrate savings of 40-60% as compared with unmodified ad-hoc routing protocol. Zhang and Cao propose DCTC [10] for target tracking in WSNs. A tree structure is constructed for moving target tracking. It uses a prediction-based schema for tree expansion and pruning. Only nodes near the target are activated. While effective for target tracking, this approach does not address topological changes to the discrete scalar field. In [11], an advanced sweep algorithm is implemented in order to activate sensor nodes in front of an event wavefront and to deactivate sensor nodes behind the wavefront. The algorithm is implemented in a small network consisting of Mica2 MOTES and demonstrates effective detection of the wavefront while conserving network resources. However, as the author states, the topological sweep algorithm does not admit a distributed approach. Duckham et al. [12] describe a triangulation approach for monitoring dynamic fields. The sensor network is triangulated and most sensors that are not in the event regions are deactivated.

In this paper, a combination of traditional clustering is used in conjunction with GAF in order to ensure network energy conservation without compromising event detection. Specifically, sensors along the boundary of different regions are activated to detect the boundary's evolution, while sensors that are in low activity regions and are not critical in terms of routing are deactivated. When coupled with our local topology discovery approach, the efficient reporting of topological change becomes feasible.

3 Event Detection in Sensor Networks

3.1 Event Detection

The goal of this research is to efficiently detect the topological changes outlined in 2.2, through distributed computations in a distributed WSN. Each of the topological changes implies that the corresponding phenomena's boundary has undergone a dynamic change. The node set approximating the boundary will also change, so long as network resolution is sufficient. It follows that those incremental changes of importance occur at the boundary or create a new boundary. Thus, changes to boundaries allow each of the topological changes to be detected. For the sake of simplicity, only incremental changes will be tested via simulation. This can be justified since: (1) the change in sensor value (relative to threshold) of a single node can result in any of the six topological changes under investigation, and (2) methods to compute non-incremental change as an extension of incremental change will be outlined.

In Figure 4, two regions, R_1 and R_2, are illustrated, along with three nodes that have changed status and must compute if a topological change has occurred. Assuming that each node stores the ID of the region it belongs to, in the best case, the outcome can be determined on the basis of the neighborhood ring and subsequent neighbor components. This is true for hole formation, hole loss, merging, and self-merging. Consider node *u*, shown in Figure 4. As a hole forms in the region monitored by the node, its sensor detects this change, and hence node *u* uses its neighborhood ring to determine the kind of change. Since its neighborhood ring is of the form [1,1,…,1], even without region ID, node *u* identifies that the change is a hole formation. Hole loss is similar, except that node *u* proceeds from a sensor value of 1 to 0. Node *v* illustrates a merge, which can be determined by the combined facts that: (1) the neighborhood ring consists of multiple neighborhood components (above threshold) and (2) the ID differs between two or more components. If region ID were the same, then the change would be a self-merge. In the case of splitting and self-splitting, region ID and neighbor data is not sufficient. Consider node *w* as an example. By observation, a self-split is taking place since node *w*'s sensor status change has not split the region into multiple components. Node *w* however, will not know this until it confers with the nodes of this region. In contrast to a merge or self-merge, prior to a split or self-split, all nodes belonging to the region will store the same region ID (R_1 in this case) regardless of the event type. So, node *w* must pass a message through the region to determine if it has an *irreducible cycle* – a broadcast cycle passing through nodes of status 1, such that the cycle can't be trivially reduced without passing through a node of status 0. In particular, a cycle discovery message is passed to one node of each neighborhood component. Each receiving node in turn passes the message to a node in each of its neighborhood components. If a node receives messages originating from different neighborhood components of the initiating node, then an irreducible cycle exists. Part of this process is illustrated by the smaller, transparent circles in the figure. In this case, Node x receives messages from different neighborhood components of Node w. Thus, an irreducible cycle has been identified and therefore, the event is a self-split. Node x of course must relay this to Node w.

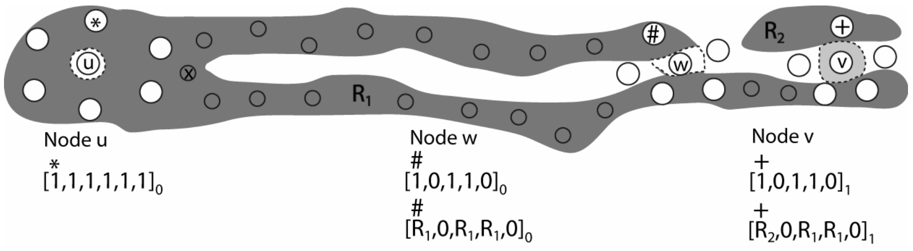


Fig. 4. One topological change from each of the three categories

This process is repeated until a node can no longer propagate the message, or a node has received multiple messages. If the latter occurs, as it would above, then an irreducible cycle exists. Thus, the topological change is a self-split. Otherwise, the change is a split.

In the case of non-incremental changes, the network is prone to non-scalable behavior and data conflict. To emphasize this, let us compare a non-incremental change without the use of assumption (5) to the same change with the use of assumption (5). We will refer to Figure 5, which displays nodes u and v that have changed status; cluster heads A, B, C ,and D; unnamed nodes; and communication links, labeled with the transmission round. A transmission round such as 2_v indicates the second round of transmission relative to node v. If there is no subscript, it indicates that the order of transmission has been coordinated by the collaborating cluster heads. A network without cluster heads is illustrated in the left pane of Figure 5. Here, nodes *u* and *v* change status, compute any topological changes, and pass the update through the component. The first two rounds of message passing are shown. Two problems arise. The first is that the number of nodes requiring updated genus data is significant: all of the nodes in the grey region. The second problem is that intermediate nodes (lying between the rippled lines) will receive conflicting data regarding the topological state of the region: node *v* reports the loss of a hole while node *u* reports the gain of a hole. By admitting assumption (5), the network is clustered as in the right pane of Figure (5). Node *u* reports a change to its cluster head *A*, which in turn reports to the affected adjacent cluster heads *B*, *C*, and *D*. Node *v* senses a change and reports this to cluster head *C*. Cluster head *C* reports one required update to node *A*, while cluster heads *B* and *D* broadcast no such report. Cluster head *A* orders the queue, first by cluster {A, C}, then by node – in this case {*u*, *v*} – and passes the data to node *u* in the third round. Node *u* makes a partial computation, passes it to cluster head *A*, which in turn passes the partial result to cluster head *C*. Cluster head *C* passes the computation to node *v*, which completes the computation, and unicasts the final result to cluster head *C*. By the eighth round, cluster head *C* multicasts the updated genus to cluster heads *A*, *B*, and *D*. The first eight rounds of message passing are shown in the figure. In order to limit network noise, the multi-channel capability of the sensor nodes is exploited: in addition to the general broadcast channel, there is a cluster-head channel, a channel for each sensor value, and four channels assigned to clusters, so that no two adjacent clusters have to operate on the same bandwidth. As a result, unsolicited nodes do not receive broadcasts, and therefore, do not waste energy on processing packets.

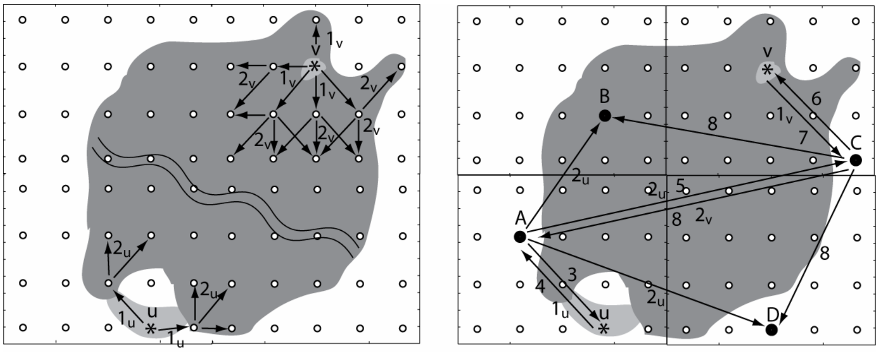


Fig. 5. A flat approach on the left and a tiered approach on the right

Since each cluster in the tiered approach behaves like a node in the flat approach, a 2-level network is "more scalable" but not truly scalable: cluster size, and the number of tiers are governed by the expected "size" of the topological events. This will be investigated in greater detail in future work in order to correlate event size with required level of network tiering.

3.2 Resource Management

Since the WSN's nodes are battery powered and have limited processing capability, economizing network resources is key. However, such economization should not undermine the network's key goal: topological event detection. To ensure a balance between event detection and resource economization, the network must be event-driven, thereby allowing for increased data resolution in areas of topological activity. While a continuous-monitoring network cannot vary its resolution or consumption of network resources, it is capable of responding to changes rapidly, since all nodes are active. The challenge in an event-driven network is to find an acceptable level of responsiveness.

In order to meet the needs outlined above, the network is tiered. We design a 2-level hierarchy network, as in the right pane of Figure 5. The upper tier network consists of cluster heads. The lower tier network is composed of all nodes in each cluster. If there are no events near a cluster, only the cluster head is active. All other nodes in the cluster can be in sleep mode, thereby conserving energy. Each pair of nodes in adjacent clusters is reachable to each other, ensured by transmission range setting.

We separate the whole network into uniform, rectangular clusters. The diagonal of each cluster is less than half the radio range in order to ensure each pair of nodes in adjacent clusters has the ability to communicate with each other. The clusters are fixed after network initialization. Assuming nodes are GPS-equipped, they can compute their geographic position, and therefore, the cluster they lie in. Each cluster has a designated cluster head, which is responsible for node activation. A node can deactivate itself if it does not detect any interesting activity.

Each node in the network can be in three states. In the sleeping state, a node does not send or receive any messages from others. This brings the most power saving to the network. In the listening state, a node can receive its neighbors' messages to determine if it should turn itself to active state but does not transmit messages. In the active state, a node collaborates with other active neighbors to monitor physical events and changes. The state transition graph of a node is shown in Figure 6. In Figure 6: (0) A node boots up. (1) A node periodically changes from active to listening mode, if there is no reading difference between the node and its neighbors over a time t_1. (2) A node periodically changes from listening to sleeping mode after a predefined time t_2. (3) A node changes to active state after it receives an activate message. (4) A node goes to listening after sleeping for a time t_3. More details about the node activation and deactivation will be explained in section 4.2.

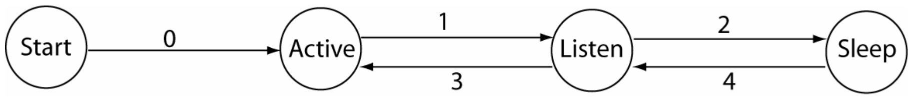


Fig. 6. State change graph

4 Algorithms

Three algorithms will be presented, covering: cluster head election, node management, and topological event detection.

4.1 Cluster Head Election

We use a randomized algorithm to select cluster heads. Each node in a cluster randomizes a short timer [13]. After the time out, if the node does not receive a suppression message from a cluster head, it becomes the cluster head and broadcasts a suppression message to inform all other nodes in the cluster. Since the cluster head is the only activated node in a cluster, its battery will discharge more quickly than other nodes' batteries. Thus, we promote a new cluster head after the battery power is lower than a designated threshold. If a cluster head's power drops below a threshold value, it periodically sends a request message with its residual energy information to all neighbors in the cluster. All nodes in listening state can receive this message. After receiving the message, a node that has more energy will randomize a short timer. After the time out, if it does not receive a new suppression message from other nodes, it becomes the new cluster head and broadcast a suppression message to all other nodes in the cluster.

4.2 Node Activation/Deactivation

Each node in a cluster is activated by the cluster head and it is self-deactivated based on the reading difference. At initialization, all nodes are in active state. Each cluster head also receives readings from, and broadcasts readings to, neighboring cluster heads. If they are all above - or below - the threshold value, no change is detected and the corresponding cluster nodes will not be activated. If a neighboring cluster head has a different Boolean response relative to threshold, then an event boundary must exist between them. In this case, the cluster head broadcasts messages to activate all nodes in its cluster. Each node, after entering the listening mode, can receive the activation request and turn itself to active state. A node in the active state will periodically compare its reading to neighbors. If there are some differences between them, it remains active for event detection. Otherwise, if there is no difference (no change) after a user specified period, it enters to listening mode for a while. If it does not receive further activation messages, it proceeds to sleep state, and then periodically alternates between listen and sleep states. The algorithm for node activation and deactivation is shown in algorithm 1.

```
While (1)
{
  if (in sleeping mode)
     After a short time $T_3$, go to listening state;
  else if (in listening mode)
     if (receives an active message)
        Go to active state;
     else
        Go to sleeping state after a short time $T_2$;
  else if (in active mode)
     if (no reading differences between any neighbors)
        Go to listening mode after a short time $T_1$;
}
```

Algorithm 1. Non-cluster node activation and deactivation

4.3 Event Detection

In this section, we discuss the algorithm for event detection. If a node changes status with respect to threshold, it tests for the easiest topological changes first, and then proceeds in order of difficulty. If the neighborhood ring is uniform, then the topological change is either hole loss or hole formation. If however, the neighborhood ring is not uniform, the node checks first for non-topological changes (not outlined in this paper), and then for the topological changes merge and self-merge. If neither event is possible, the node broadcasts a message in order to discover any irreducible cycles. If there are such cycles, the event is a self-split, and otherwise, the event is a split. It should be noted that the algorithm is written for nodes that do not lie on the boundary of the network. While the modifications necessary to include boundary nodes are reasonable, they are not included below.

```
While (1)
{
  if (sensor status changes)
     Check neighborhood ring;
     if (uniform)
        if (sensor status of node is 1)
           Event: hole loss
        else
           Event: hole formation
     elseif (one neighborhood component)
           Non-topological event
     else
        Check neighborhood components' region IDs
        if (multiple region IDs)
           Event: merge
        else if (one region ID)
           if (sensor status of node is 1)
              Event: self-merge
           else
              Check for irreducible cycles
              if (irreducible cycles)
                 Event: self-split
              else
                 Event: split
}
```

Algorithm 2. Event detection algorithm

5 Simulation and Discussion of Results

5.1 Simulation

We evaluated our algorithms using NS2 [14], open source, network simulation software. NS2 contains standard API that facilitates the development of a network model at the network level, the node level, and the process level. In addition, many research groups have made their custom node and network models available, thereby expediting the development of future work, such as ours. In the simulation, a 100m by 80m WSN that consists of 400 nodes with 20 clusters is constructed. Each cluster is a 20m by 20m grid and there are 20 evenly distributed nodes in each. As specified in section 3, we assume nodes in adjacent clusters can communicate directly. In order to achieve this assumption, the node communication radius is set to 57m. Each node saves its nearest 8 neighbors' information around it in the neighborhood ring. The event-detection algorithm is applied in conjunction with both continuous and event-driven monitoring approaches. In the case of continuous monitoring, each node is active throughout the entire simulation interval. As discussed in the related work, even an idle node consumes a good deal of energy. Hence, the greatest savings only result from deactivating nodes. Thus, to measure network savings, it suffices to determine, and compare, the percentage of active nodes between the two approaches. Furthermore, the time taken to detect topological change is computed as a metric for network responsiveness.

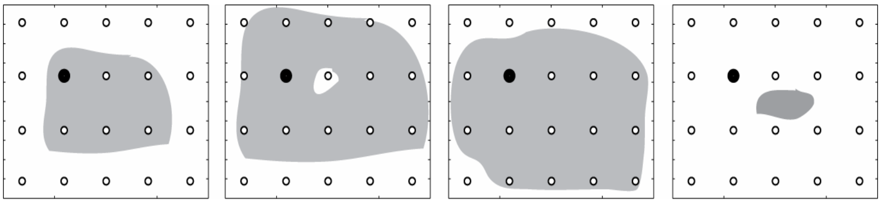


Fig. 7. One cluster at t=10s, 20s, 30s, and 45s

Table 1. Active node count at t=10s, 20s, 30s, and 45s

	Continuous approach	**Event-driven approach**
T=10s	400	191
T=20s	400	191
T=30s	400	191
T=45s	400	20

Over the course of a single simulation run, the network is initialized at t=1s. An areal object forms in the middle of the field at t=2s and continues to grow. A hole emerges in the areal object at t = 20s, and grows until t=25s, when it begins to shrink. At t=30s, the hole disappears. Then, the areal object continues to shrink and disappears at t=44s. In this scenario, sensor nodes that are active sample the environment once per second. From the simulation results, the areal object which forms at t=2s is detected at t=4s, after it covers a cluster head. After detecting the areal object, the cluster head activates all nodes in its cluster and notifies adjacent cluster heads. Other topological changes, the hole formation at t=20s, the hole loss at t=30s, and the disappearance of the areal object at t=44s are all detected immediately. In the continuous approach, all topological changes are detected immediately after they happen. We also compare the number of active nodes between the continuous and event-driven approaches at some time snapshots t=10s, 20s, 30s, and 45s. The results are shown in Table 1. Clearly, the continuous monitoring approach keeps all nodes in active state, but the event-driven approach keeps only a few nodes in active state all the time, which means larger energy savings for the entire network.

5.2 Discussion

In the simulation, all topological changes are detected correctly in the network, which means our event detection algorithms works for both event-driven approach and continuous approach. Compared to previous base station based approaches [4, 11], such in-network detection is one of the most significant contributions of our work. Node will report changes to the base station which need no future process at the base station side.

By the event-driven approach, as indicated by the simulation results, the areal object appears at t=2s, but it was not detected before t=4s when the areal object covers a cluster head, which means the event-driven approach may decrease the responsiveness of the network. In comparison, the continuous approach allows for instant detection at the cost of network resources. While an event-driven approach is necessary to handle lengthy, remote deployments, it is inevitable that energy conservation sacrifices resolution and responsiveness to some degree. In particular, when a topological change occurs in a cluster that has been set to sleep, it will go unnoticed until the cluster head itself detects it. Clearly, as the cluster size decreases, responsiveness and resolution increase. However, network resources would be compromised, as in the continuous monitoring case. There should be a trade-off between responsiveness and the energy consumption. If we need better responsiveness, we can set small cluster size. If the responsiveness is not very crucial, a larger cluster size could be used. Bounds on cluster size relative to event detection will be addressed more thoroughly in future work.

6 Conclusion and Future Work

This paper presents algorithms for topological change detection in a WSN using event driven approaches. Different from previous approaches, we focus on detecting topological changes of areal objects monitored by the WSN. A neighborhood ring data structure is proposed for in-network event detection. By our event detection algorithms, topological changes can be detected directly in the network, other than computed at the base station after receiving all reporting messages. Each node does not have to send readings to the base station for processing. This characteristic is one of the most significant differences compared to traditional approaches. By our event-driven approach for network management, not all nodes are required to be in the active mode all the time and then the network energy is saved greatly. The simulation results show that our event-driven approach deactivates some nodes in the network without decrease responsiveness significantly. The event detection algorithms proposed also detect all topological changes correctly in the network.

Our current simulation does not include split and self-split and the detection of such changes need global broadcasting using current algorithms via the irreducible cycles. In future work, the topological changes split and self-split will be addressed and we are trying to reduce such global broadcasting. Furthermore, non-incremental changes need to be simulated to confirm that a tiered approach effectively handles more complicated, non-atomic topological changes. This would include the correct assignment of a component's ID, as well as passing off partial computations between clusters.

Acknowledgements. This material is based upon the work supported by the US National Science Foundation under Grant Nos. IIS-0534429 and IIS-0429644.

References

1. Jiang, J., Worboys, M.: Event-based topology for dynamic planar areal Object. Technical report, University of Maine (2007)
2. Cheng, X., Thaeler, A., Xue, G., Chen, D.: TPS: A time-based positioning scheme for outdoor wireless sensor networks. In: INFOCOM, pp. 268–2696 (2004)
3. Shang, Y., Ruml, W., Zhang, Y., Fromherz, M.P.J.: Localization from mere connectivity. In: MobiHoc, pp. 201–212 (2003)

4. Funke, S.: Topological Hole Detection in Wireless Sensor Networks and its Applications. In: Proceedings of 3rd ACM/SIGMOBILE International Workshop on Foundations of Mobile Computing (DIALM-POMC), Cologne (2005)
5. Fang, Q., Gao, J., Guibas, L.: Locating and Bypassing Holes in Sensor Networks. Mobile Networks and Applications 11, 187–200 (2006)
6. Chintalapudi, K., Govindan, R.: Localized edge detection in sensor fields. Ad Hoc Networks 1(2-3), 273–291 (2003)
7. Ding, M., Chen, D., Xing, K., Cheng, X.: Localized fault-tolerant event boundary detection in sensor networks. In: INFOCOM, pp. 902–913 (2005)
8. Chen, B., Jamieson, K., Balakrishnan, H., Morris, R.S.: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In: Proceedings of the ACM/IEEE international conference on Mobile Computing and Networking (July 2001)
9. Xu, Y., Heidemann, J., Estrin, D.: Geography-informed Energy Conservation for Ad-hoc Routing. In: Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 70–84 (2001)
10. Zhang, W., Cao, G.: DCTC: Dynamic Convoy Tree-Based Collaboration for Mobile Target Tracking. IEEE Transactions on Wireless Communications 3(5), 1689–1701 (2004)
11. Liu, J., Cheung, P., Guibas, L., Zhao, F.: A Dual-Space Approach to Tracking and Sensor Management in Wireless Sensor Networks. In: WSNA, pp. 131–139 (2002)
12. Duckham, M., Nittel, S., Worboys, M.: Monitoring dynamic spatial fields using responsive geosensor network. In: Proceedings of the 13th annual ACM international workshop on Geographic information systems, pp. 51–60 (2005)
13. Meng, X., Li, L., Nandagopal, T., Lu, S.: Event contour: an efficient and robust mechanism for tasks in sensor networks. Technical Report, UCLA (2004)
14. The Network Simulator – NS2, `http://www.isi.edu/nsnam/ns`

Uncovering Hidden Spatial Patterns by Hidden Markov Model

Ruihong Huang and Christina Kennedy

Department of Geography, Planning & Recreation
Northern Arizona University, Box 15016
Flagstaff, AZ 86011-5016, USA
{Ruihong.Huang,Tina.Kennedy}@nau.edu

Abstract. Many spatial data mining and spatial modeling approaches use Euclidean distance in modeling spatial dependence. Although meaningful and convenient, Euclidean distance has weaknesses. These include providing an over simplified representation of spatial dependence, being limited to certain spatial pattern and symmetrical relationships, being unable to account for cross-class dependencies, and unable to work with categorical especially multinomial data. This paper introduces Hidden Markov Model (HMM) as an attractive approach to uncovering hidden spatial patterns. The HMM assumes that a hidden state (factor or process) generates observable symbols (indicators). This doubly embedded stochastic approach uncovers hidden states based on observed symbol sequences using two integrated sets of probabilities, transition probability and emission probability. As an alternative to Euclidean distance based approaches, the HMM measures spatial dependency by transition probabilities and cross-class correlation better capturing geographic context. HMM works with data of any measurement scale and dimension. To demonstrate the method, we assume urban spatial structure as a hidden spatial factor underlying single family housing unit prices in Milwaukee, Wisconsin, we then use the HMM to uncover four hidden spatial states from home sale prices.

Keywords: GIS, Hidden Markov Model, spatial modeling, data mining.

1 Background

Many spatial pattern discovering approaches use Euclidean distance in modeling spatial dependency. Spatial clustering, represented by k-means clustering [17][11][44] and its variations such as k-medoid [20] and Expectation Maximization (EM) algorithm [8] cluster objects based on the distance between them [17]. Clustering using Euclidean distance suffers from two difficulties. First, if attributes of spatial objects are not considered, Euclidean distance is the sole surrogate for spatial dependency. Such methods can find only spherical-shaped clusters and encounter difficulty discovering clusters of arbitrary shapes [17]. Second, if attributes are of concern, distance between objects is usually computed based on subjective weights of attributes. With multi-dimensional data, any subjective weighting system can be problematic. With categorical data, weighting distance becomes impossible.

The same issues exist in other clustering and hot spot detecting approaches that use Euclidean distance. For example, the dendrogram-based clustering methods including AGNES and DIANA [20] [17] and their variations split or merge objects also based on Euclidean distance. Furthermore, hot spot detection approaches such as G and G* autocorrelation statistics [14] [34] and spatial statistical approach including geographically-weighted regression [4] require some form of a distance weight matrix which is typically calculated from the Euclidean plane. A weight matrix carries with it the assumptions and limitations of the Euclidean model [30].

The first law of geography [47] has been exploited by spatial modeling techniques [29]. This is best illustrated by the Inverse Distance Weighted (IDW) method in which the influence of an event on a prediction location is inversely related to the distance between an observed location and the prediction location. Euclidean distance is the sole proxy of spatial interactions in this model. Although convenient and meaningful, Euclidean space is not the only representation of geographic space since some geographic processes can have non-Euclidean properties [30]. For example, human-made or natural networks often channel spatial interaction so that the shortest path between two locations may no longer correspond to a straight-line segment. In other instances such as the evolution of weather systems, geographic context (the initial conditions) is critical. Moreover, human systems tend to exhibit sensitivity to geographic context [30] and therefore cannot be completely accounted for by Euclidean distance. Cliff and Haggett [7] argue that the Euclidean plane is limited for analyzing spatial diffusion processes such as the spread of disease over geographic space. Based solely on Euclidean distance, the IDW usually produces circular patterns and even bull's eyes in modeling results. In addition, this deterministic method does not provide estimate of uncertainty of output.

Geostatistical modeling approaches represented by Kriging provide estimates of the uncertainties in results. Although spatial dependency is measured by more sophisticated indicators, including semivariogram or covariance, the predictor of Kriging also uses Euclidean distance. While Kriging models have been widely used in applications with continuous variables and with cases using discretized continuous variables in indicator Kriging [3][15][52], they have limitations in handling categorical variables [26]. Moreover, all conventional geostatistical methods have difficulties accounting for interclass dependencies which include not only cross-correlations as measured by indicator cross-variograms but also the juxtaposition relationships in spatial distribution of multinomial classes and directional asymmetry [25][53]. Because of the intrinsic symmetric property of indicator cross-variograms, many variogram-based models are not effective in capturing the directional juxtaposition relationships.

Unlike all the approaches discussed above, Markov cross transition probabilities have the capability of representing the interdependencies of classes [26] and avoiding the Euclidean pitfalls. Traditionally, a Markov chain is a discrete-time, stochastic process in which the next state depends on the present state. When applied to spatial problems, a one-dimensional Markov chain can be established on regularly spaced sampling locations along a survey line. A one-dimensional Markov chain can be expanded to a Markov field by including an additional dimension of state transition probabilities [10][53].

In spite of its strength in representing interdependencies between classes, capabilities of working with of both discrete and continuous variables, as well as convenience of handling multi-dimensional data, a traditional Markov model makes no inference about underlying processes that generate the observed phenomena. For many geographic problems, the controlling factors and processes can be hidden from direct observation. In this paper we introduce the Hidden Markov Model (HMM) as a new method for uncovering hidden spatial patterns based on observed indicators. We discuss key concepts and principles of HMM in section 2; develop solutions for applying the method to a spatial problem, to uncover urban spatial structure from home prices, in section 3; introduce the case study area in section 4 and present modeling results in section 5. Finally, we conclude the study in section 6.

2 Hidden Markov Model

The Hidden Markov Model (HMM) is a doubly embedded stochastic method based on probability theory. It was initially studied in statistics in the 1960s and early 1970s but has gained popularity with advancements in artificial intelligence (AI) since the 1980s [42]. Following its great success in speech recognition, it has been applied to a wide range of pattern recognition and data mining applications such as gesture recognition [45], handwriting recognition [33], recognition by robots [1], vegetation dynamics analysis in remote sensing [49], precipitation occurrence pattern [18][43], and spatiotemporal pattern of land use change [28].

The principle of information retrieval from observed signals by HMM is best illustrated by speech recognition. A sentence consists of a sequence of words. Though the machine does not really understand spoken words since what it ‘hears’ are merely vibration signals, the words behind the signals can be discovered based on a doubly embedded stochastic process with one unobservable [42]. The first probability is that of guessing each individual word from a segment in a series of signals. Since a piece of signal is pronounced from a word, the machine can be trained to guess a word based on the signals with a probability though a one-to-one relationship between words and signals is impossible due to uncertainties in homonyms, voices, tones, accents and so on. In turn, a piece of signal is only one of many possible manifestations or emissions of the word. The probability of emitting a particular signal from a word is known as *emission probability*. The second probability governs the word-to-word transition in a sentence. For example, a string of signals could be segmented and interpreted as words “I – love – you” or “eye – love – you” based on emission probabilities. However when *transition probability* is considered, the later interpretation can be rejected because, based on the training the computer has received, the probabilities of transitioning from “eye” to “love” is zero, but the former sentence can be accepted because the probability of transitioning from word to word is the greatest among all possible guesses (Figure 1).

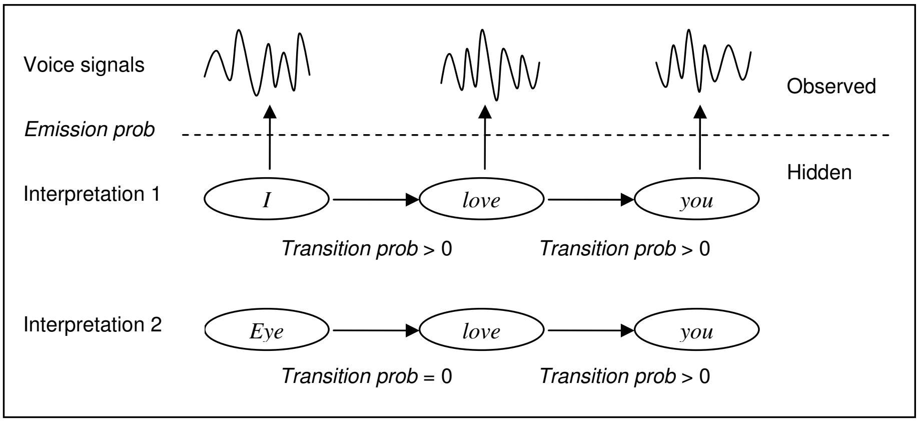


Fig. 1. An example of speech recognition by HMM

In the above speech example, the hidden vocabularies are named *states*, and the observed signals are named *observation symbols*.

Similarly, we posit that an HMM can be used to uncover the impact of relative location and urban spatial structure. We assume that location within the city is a hidden factor underlying the general trend of housing price while the intricate structural externalities add variations to actual home sale prices. This hidden spatial factor can be discovered from home sale records by using Hidden Markov Models. The hidden factor is represented by *spatial states* which can be equated to vocabularies and the observed symbols (frequency distribution of house prices) are *signals* emitted by corresponding states. The following will facilitate understanding specifics of how HMM functions.

(1) Assuming there are N individual states in a Markov system, we denote the set of states as $S = \{S_1, S_2, \ldots, S_N\}$. In an *ergodic* model any state may be reached from any other with a *transition probability*. For example, if urban neighborhoods are the states in question, then there is a probability for one type of neighborhood to abut another type.

(2) The probability of transitioning from state i at time t to state j at time t+1 is

$$a_{ij} = P(q_{t+1} = S_j \mid q_t = S_i),\ 1 \le i, j \le N \tag{1}$$

where q_{t+1} and q_t are the state variable at time t+1 and t respectively. In an HMM, all transition probabilities constitute a matrix $A = \{a_{ij}\}$.

(3) Observation symbols such as acoustic signals are assumed to be emitted by hidden state. The set of all observable discrete symbols in an HMM is denoted as $V = \{v_1, v_2, \ldots, v_M\}$. Discrete symbols can be derived from training processes. Observations can also be continuous values.

(4) With a discrete symbol system, the probability for symbol k to occur in state j at time t is

$$b_j(k) = P(v_k \text{ at } t \mid q_t = S_j),\ 1 \le j \le N \text{ and } 1 \le k \le M \tag{2}$$

where v_k is the observed symbol and q_t is the state variable at time t. This probability is the *emission probability* or *bias* since an observation is regarded as a biased representation of a state. All emission probabilities in an HMM are denoted as $B = \{b_j(k)\}$. Emission probabilities are also derived from training.

With continuous observation values, the emission probability density of a value can be calculated by using a *probability density function* (pdf) established from a training dataset. In the speech recognition example, a probability density function can be a mixture of multiple kernel (usually Gaussian) functions. If a state (vocabulary) has M mixtures, the probability density of emitting observation o_t in state j is

$$b_j(o_t) = \sum_{m=1}^{M} c_{jm} f(o_t, \mu_{jm}, U_{jm}) \tag{3}$$

where c_{jm} is the coefficient of m-th mixture in state j such that $\sum_{m=1}^{M} c_{jm} = 1$, and f is the kernel density function with mean μ_{jm} and covariance matrix U_{jm} for the m-th mixture component in state j [42][32].

(5) Finally, an *observation sequence* (O) consisting of T observations is denoted as $O = \{o_1, o_2, \ldots, o_T\}$. At the initial time ($t = 1$) every state has a probability of occurring – the *initial probability*. For state i, the initial probability is denoted as $\pi_i = P(q_1 = s_i)$. The set of all initial probabilities in an HMM is denoted as $\pi = \{\pi_i\}$.

For convenience, a hidden Markov model is usually denoted as $\lambda = \{A, B, \pi\}$. Generally, hidden Markov models address the following problems [42]:

Problem 1: given an HMM $\lambda = \{A, B, \pi\}$ and an observation sequence $O = \{O_1, O_2, \ldots O_T\}$, what is the probability of the observation sequence $P(O|\lambda)$?
Problem 2: given an HMM $\lambda = \{A, B, \pi\}$ and an observation sequence $O = \{O_1, O_2, \ldots O_T\}$, what is the most probable sequence of states underlying O?
Problem 3: given λ, adjust the parameters of the model $\lambda = (A, B, \pi)$ to maximize $P(O \mid \lambda)$.

Problem 1 is a typical gambling problem. It evaluates how an HMM predicts outcomes. The forward-backward algorithm provides an efficient solution to this problem. Since predicting observations is not the focus of this paper interested readers are referred to [42] and [32] for details. Problem 2 is more interesting to us as it discovers the hidden states that generate the observations, so that we may use it to uncover the spatial states that underlie housing prices. This problem can be solved by the *Viterbi algorithm* (Appendix A). The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states, called the *Viterbi path*, that produces a sequence of observed events given a hidden Markov model[1].

Problem 2 requires an HMM. This HMM can be provided by solving problem 3 which trains HMMs based on observation data sets. Problem 3 is solved by a k-means

[1] In Appendix A, formulas 10–13 transfer the problem of maximizing the probability of an observation sequence with a corresponding state sequence into a problem of minimizing the cost of an optimal state sequence. By recursively solving the minimal cost of every step in the observation sequence by formulas 14–19, a Viterbi path, *i.e.* state sequence with minimal cost or maximal probability, can be retrieved by formula 20–21.

training algorithm. If the states in a training dataset are known, a supervised training is conducted to compute the three HMM probability sets. In preparing a training data set, for example, a training sentence can be read by different speakers. In this case, the words in the training sentence are known states. In most situations, such as training with acoustic signals recorded from conversations in a natural noisy environment, the words are unknown and are usually connected to one another in sentences. Therefore unsupervised training is required. K-means unsupervised clustering and training can achieve 98 to 99 percent of recognition string accuracies [41]. The existence of unsupervised training algorithms that allow for estimation of model parameters from a body of observations is a major advantage of HMM [28]. After clusters are identified in the training data set, they are used as estimates of states. Finally, HMM parameters are estimated by the following formulas [42]:

$$\overline{\pi}_i = \text{expected frequency (number of times) in state } i \text{ at time } (t = 1) \tag{4}$$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i} \tag{5}$$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j} \tag{6}$$

3 Uncovering Urban Spatial Structure by HMM

The HMM should be attractive to geographers. It was, however, first applied to spatial problems by computer scientists to discover spatiotemporal patterns of land use change from remotely sensed images [28]. In order to make the HMM more versatile and capable of working with various spatial data dimensions and measurement scales, several technical issues including symbolization, symbol sequencing, and data continuity handling need to be addressed. We explore these issues by applying the method to the discovery of spatial structure from observed housing prices in an urban space.

Empirical research on housing prices using hedonic models has displayed contradictory and spurious results if locational externalities such as proximity to certain land uses are included [35]. While technical issues, such as the measurement of locational externalities (e.g. arbitrary thresholds for proximity) contribute to the controversy [5], the uncertain nature of locational variables on housing price may be regarded as the root of blame [35][6]. Instead of breaking the impact of location down into a number of externalities as do hedonic models, we assume that relative location in an urban space, as a whole, is a hidden factor controlling housing prices. By studying spatial characteristics of housing prices, we can learn the impact of location and then discover the spatial structure of an urban area.

The above assumption is based on the following justifications: First, it is common knowledge that houses of similar size and quality can differ greatly in price for different locations. Second, relative location within an urban area, as represented by closeness or adjacency to amenities or unfavorable land uses, does have impact on housing price [6][16]. Third, features sharing a geographic space display similarities since they tend to be influenced by common external processes and spatial interactions

[36]. Common physical and socioeconomic processes in urban neighborhoods lead to the widely observed clustering and segregation in urban housing [38][19][50][13] as well as spatial autocorrelation in housing price [9][2].

Relative locations in an urban area are indicated by urban zones. However, the boundaries between urban zones are usually obscure and often are determined on subjective criteria. For example, in examining the evolution of the City of Milwaukee's spatial structure, various distinctive sub-areas were subjectively defined based on the location, demographic composition, and socioeconomic status of aggregation of census tracts [37][48]. Moreover, a diversified array of criteria including subdivisions, major streets, physical and natural barriers, housing style and etc. were used to determine boundaries of the urban neighborhoods in Milwaukee [31]. The HMM provides an objective alternative to the discovery of urban spatial structure using observed housing prices within the City of Milwaukee as a case study. The overall process of spatial state discovery is illustrated in Figure 2.

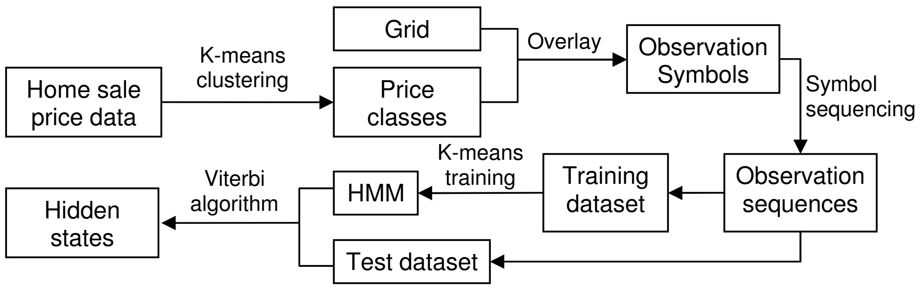


Fig. 2. Workflow of spatial state discovery by HMM

3.1 Observation Symbolization

Defining observation symbols is the first step of HMM data processing. Instead of using individual home sale prices, we take the price frequency distribution of spatial units as observation symbols in this study. In doing so, we first define spatial units by partitioning the study area into grid cells. In order to meet sample size requirements, traditional quadrat analysis recommends a cell size of $2A/r$ [51], where A is the area of the study region and r is the total number of observation locations. In this study, we adopted a cell size $2Ad/r$ for multi-dimensional symbols, where d is the symbol dimensions, so that sample size is satisfied for each symbol dimension as well. With 6048 data points falling within about 600 km^2 of the study area and d = 5, a grid cell size of 1 km x 1 km is chosen in this study. This way, the 600 km^2 study area is partitioned into 30 rows by 20 columns.

We then symbolize each spatial unit by using a frequency distribution (e.g. histogram) of home sale prices within the unit expressed as a number of percentage values for various price classes. Suppose a spatial unit contains a number of home sale data points of which 5% falls in very low price category, 20% in low price, 50% in medium, 20% in high, and 5% in very high price category, the spatial unit is symbolized as a vector (5, 20, 50, 20, 5). In this example, each percentage value in the symbol is referred to as a dimension. This symbol describes structure of the housing prices within the spatial unit.

If a dataset consists of one continuous variable such as housing price, symbol dimensions can be defined by a data categorization approach. We chose a k-means clustering to create five house price categories, i.e. five symbol dimensions, in this study. As in ordinary thematic mapping, we use five categories because fewer may not capture enough detail of price structures; but, too many would require increasing the size of spatial units in order to satisfy sample size requirements and would increase computation complexity. K-means clustering is used because it is an efficient unsupervised method that produces tight classes by minimizing differences within classes and maximizing differences between classes. Since this k-means clustering is used simply to categorize the non-spatial house price data, the spherical spatial pattern constraint [17] does not affect results. However, because of the heuristic characteristic of k-means algorithms, initial means may affect the results of clustering. After comparing four popularly used initialization methods, Pena *et al* [39] conclude that both random and Kaufman initialization are effective methods and are more independent of initial clustering and instance order than Forgy and MacQueen. To ensure quick and consistent clustering, we first categorize the data set into quantiles, then randomly select a value as an initial cluster mean from each quantile. We find that this cluster initialization method can quickly converge clusters to stable conditions.

3.2 Symbol Sequencing

Symbol sequencing defines Markov chains of observation symbols. It is not an issue for time serial observations such as in speech recognition since a sequence is inherent in the observation process. Defining observation sequences in spatial data can be a challenge especially for random location observations. Common practices include regular-spacing sampling and space partitioning. For example, Zhang and Li [53] define Markov chains by sampling land cover type with a regularly spaced lattice over a vector map; Elfeki and Dekking [10] regularly sample a 200 km long and 50 m deep geologic cross-section and establish coupled Markov chains that integrate the horizontal and vertical transition probabilities; Tjelmeland and Besag [46] propose a hexagonal array to determine the first- and second-order neighborhood in studying Markov random fields; Lovell [27] defines a discrete concentric circular search space within images in feature recognition; and Mari and Le Ber [28] present a more complex sampling method, the fractal Hilbert-Peano curve sampling within satellite images, in a spatial land use change study.

In this study, we partition the space into 30 x 20 grid cells of 1 x 1 kilometer in size. For efficiency, we define two observation sequences by serializing the grid by alternate advancing technique [26], one in horizontal and the other vertical direction. Once observation sequences are created for a training dataset, a k-means training procedure is performed to compute the probability sets of an HMM (π, A, B) from the sequences by using equations 4 through 6. Prior to computation of the probability sets, an unsupervised k-meanings training algorithm defines states by k-means clustering of observation symbols. In order to uncover spatial states by the Viterbi algorithm, a test dataset consisting of observation sequences is prepared with the same procedure.

3.3 Hidden Spatial States to Uncover

Hidden Markov Models allow the user to specify the number of hidden states to uncover. Since we define a spatial state as an indicator of the controlling effect of location on housing price, we specify four hidden spatial states to discover in accord with the four urban zones in the city: poor inner city, affluent suburbs, well-off transition zone between inner city and suburbs, and extremely wealthy lakefront. In addition, since the study area contains a significant number of no-data spatial units, a no-data spatial state is included. Thus, we have a total of five spatial states to be uncovered.

3.4 Data Continuity

An observation symbol can be either discrete or continuous depending on the data type of its dimensions. With HMM, a continuous symbol can be converted to a discrete symbol by categorizing dimension values into discrete classes. For example, we can treat each integer percentage value as a discrete value. Therefore, an observation consisting of five dimensions would have millions of possible unique symbols. Because of the finite size of the training data set in practice, however, a manageable number of unique symbols can be derived from training. In this study, for example, we identified 620 unique symbols based on the 1999-2003 home sale data.

With continuous data handling, each dimension of a symbol takes a real number. Since the number of unique symbol is, in fact, unlimited, emission probabilities cannot be calculated by formula (6). Instead, emission probabilities have to be calculated by probability density functions. If a normal distribution is assumed for every dimension of the training dataset, the general mixture probability density function formula (3) can be simplified as a single kernel density function. For state *j* to produce observation o_t (with *D* dimensions) at time *t*, the probability density is computed as

$$P_j(o_t) = \prod_{d=1}^{D} f(o_{td}, \mu_{jd}, \sigma_{jd}) \tag{7}$$

where *f* is the kernel function, o_{td} is the *d*-th dimension of observation at time *t*, μ_{jd} is the mean and σ_{jd} is the standard deviation of the distribution. We adopt a standardized Gaussian function for the kernel probability densities, i.e.

$$f(o_{td}, \mu_{jd}, \sigma_{jd}) = e^{(-(\frac{x-\mu_{jd}}{\sigma_{jd}})^2/2)} / \sqrt{2\pi} \tag{8}$$

where μ_{jd} and σ_{jd} are estimated by the sample mean and standard deviation of the training dataset.

This study adopts both *discrete* and *continuous* data handling. Results derived from both continuity handlings are compared in section 5.

3.5 Discovering Hidden Spatial States

The Viterbi algorithm is used to discover the hidden spatial state because of its successful use in a wide range of artificial intelligence applications. With a trained HMM for the study area, the Viterbi algorithm is conducted on a test dataset to uncover hidden state sequences. Based on a selected data continuity handling, emission probabilities $b_j(O_t)$ in formulas (14) and (16) are computed from the test dataset differently.

With continuous data, the emission probability of any observation symbol can be computed by formula (7). With discrete data, an emission probability is looked up in the trained HMM by given a state and an observation symbol. However, a symbol in the test dataset may not find an identical match in the HMM because discrete symbols derived from the training process are not exhaustive. We match an observed symbol to the closest trained symbol by the following formula.

$$x = \arg\min_{1 \le i \le M} \sqrt{\sum_{d=1}^{D} (o_d - s_{id})^2} \tag{9}$$

where x is the index of the matched training symbol, o_d the d-th dimension of the observation in the test dataset, s_{id} the d-th dimension of the i-th training symbol in the HMM, M the total number of all training symbols, and D the number of symbol dimensions.

4 The Study Area

We chose the City of Milwaukee in Wisconsin as our case study because the city presents strong social and economic stratification which is reflected by distinct urban zones. We assume that relative location within an urban space is the key variable affecting housing price. We seek to uncover the underlying factor by the Hidden Markov Model and further to reveal the urban spatial structure.

The Milwaukee metropolitan area is located in southeastern Wisconsin fronting the Lake Michigan. In 2000, the population of the metropolitan area was 1,500,741 of which the city of Milwaukee accounted for 593,920 (U.S. Census). According to the Center for Economic Development at the University of Wisconsin at Milwaukee, the metropolitan area has experienced significant 'hollow-out' [21][23], polarization [22], spatial mismatch [40], and racial segregation [24].

The metropolitan area displays an apparent modified concentric zone structure from the center to periphery: a prosperous Central Business District (CBD), two (north and south) poor inner city areas, and affluent suburbs. A well-off transition zone between the inner city and suburb is clearly observable. Additionally, an extremely wealthy lakefront neighborhood lies north of the CBD (Figure 3). The apparent spatial differentiation makes Milwaukee an ideal site for testing HMM as a method for identifying underlying spatial states in urban areas.

The City of Milwaukee has an excellently maintained property database, the Milwaukee Master Property Database (MPROP), which has records of nearly 160,000 properties within the city for the past thirty years. Each property is described by more than eighty attributes covering many aspects of the land and construction upon it. Most importantly, the database maintains records of every property deed from which the transaction prices can be retrieved. All property deeds of the previous year are documented in the data files of the current year. Although the sale price of a home is not recorded directly, it can be derived from the documented convey fee that has remained constant at 0.3 percent of the sales price since 1983.

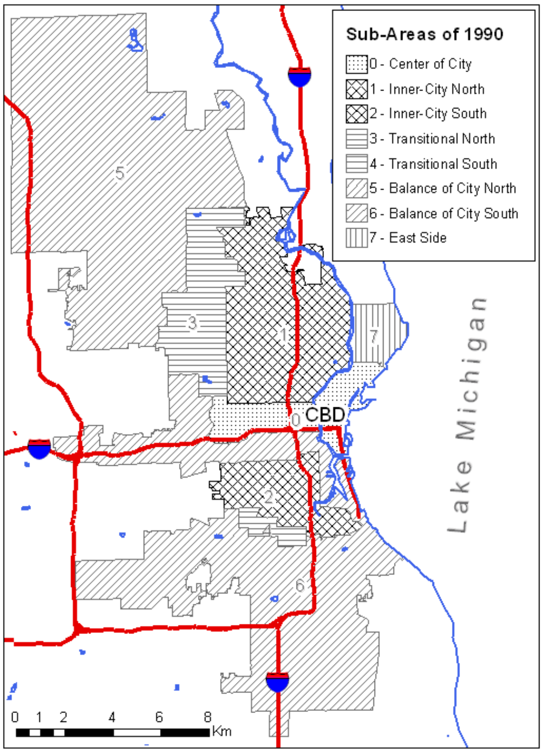


Fig. 3. Spatial structure of the City of Milwaukee (Based on [48])

This paper focuses on single family home sales since they directly reflect the market value of properties. The MPROP distinguishes 36 different types of property deeds. Since some deeds, such as court order, are not transacted through the free market, we chose to use only transactions of warranty deed (WD), trustee deed (TD) and personal representative deed (PR) in order to obtain accurate market prices of the properties. From the 2005 MPROP database, 6127 deeds of the above types completed in 2004 were extracted, of which 6048 properties are geocoded in GIS using x, y coordinates or addresses.

5 Data and Result Analyses

Six years (1999 to 2004) of single family home sale data were extracted from the MPROP database. Each year's data is symbolized (clustered by price and grided) and serialized to make an observation sequence. The first five sequences (1999 to 2003) are used as a training dataset, and the 2004 sequence used as a test dataset for spatial state discovery. All computations, including symbolization, symbol sequencing, k-means HMM training, and Viterbi algorithm, were completed by Visual Studio. NET programs.

5.1 Trained Hidden Markov Models

By combining two symbol sequencing (horizontal and vertical) with two continuity data handling (continuous and discrete), four HMMs (A, B, π) are produced and four sequences of hidden spatial states are discovered. All HMMs have the same initial probability set

$$\pi = \{1.0, 0.0, 0.0, 0.0, 0.0\}$$

Values in this set are sequentially the initial probability of spatial states 0 to 4. The initial probabilities indicate that the first observation (top-left grid cell) is state 0 for all the training sequences.

5.1.1 Transition Probabilities

Transition probabilities between states are computed from observation-to-observation (cell-to-cell) transitions of the training dataset. Models with different data continuity handling (continuous / discrete) show no difference in transition probabilities since data continuity has no effect on state identification by k-means clustering in the training process. Although the effect is slight, the symbol sequencing method, however, does influence transition probabilities. The following are two transition probability matrices derived from the same training dataset.

$$A_H = \begin{pmatrix} 0.8876 & 0.0143 & 0.0320 & 0.0519 & 0.0114 \\ 0.0728 & 0.6962 & 0.1772 & 0.0380 & 0.0158 \\ 0.1170 & 0.1149 & 0.5872 & 0.1766 & 0.0043 \\ 0.2432 & 0.0295 & 0.1794 & 0.5061 & 0.0417 \\ 0.2727 & 0.0909 & 0.1636 & 0.2727 & 0.2000 \end{pmatrix}$$

$$A_V = \begin{pmatrix} 0.8779 & 0.0131 & 0.0297 & 0.0616 & 0.0148 \\ 0.0759 & 0.7373 & 0.1646 & 0.0158 & 0.0063 \\ 0.1404 & 0.1149 & 0.5745 & 0.1702 & 0.0000 \\ 0.2555 & 0.0147 & 0.2285 & 0.4816 & 0.0197 \\ 0.2727 & 0.0000 & 0.0545 & 0.3273 & 0.3455 \end{pmatrix}$$

Where A_H and A_V correspond to horizontal and vertical data sequencing, in which, rows 1 to 5 represent spatial states 0 to 4 respectively; so do columns 1 through 5. Values in the matrices are state-to-state transitioning probabilities. As indicated by the high values on the main diagonal (from upper left to lower right) of the matrices, spatial autocorrelation in the training datasets is apparent. When two main diagonals are compared, a higher value indicates that the clustering of the corresponding spatial state is stronger in the direction of that sequencing (horizontal or vertical). For example, the transition probability from spatial state 1 to state 1 is significantly greater in A_V (0.7373) than in A_H (0.6962), which indicates the cluster of spatial state 1 tends to be vertically oriented. State 4 has dispersed transition probabilities, which suggests that the spatial state tends to be dispersed with weak clustering. All these patterns can also be found in the resulting maps in Figure 4.

5.1.2 Emission Probabilities

An emission probability indicates the likelihood for a state to produce a certain observation symbol. Emission probabilities are not affected by symbol sequencing methods but are calculated and presented differently for different data continuity handling. With continuous data handling, we use a Gaussian probability density function (pdf) for each symbol dimension of each state. Parameters of the Gaussian functions derived from the *k*-means training algorithm are shown in the matrices below.

$$\mu_{jd} = \begin{bmatrix} 0.0000 & 0.0097 & 0.0188 & 0.0325 & 0.1672 \\ 78.3101 & 17.5443 & 3.2405 & 0.4367 & 0.4715 \\ 11.6617 & 69.4638 & 18.4043 & 0.4383 & 0.0213 \\ 2.0491 & 17.5283 & 76.4791 & 3.7125 & 0.2703 \\ 4.9455 & 3.3455 & 17.6727 & 63.3636 & 10.6545 \end{bmatrix}$$

$$\sigma_{jd} = \begin{bmatrix} 0.0000 & 0.4061 & 0.7884 & 0.6931 & 3.6708 \\ 17.3252 & 15.6195 & 8.3650 & 2.7704 & 4.0335 \\ 13.3069 & 17.0824 & 16.8705 & 2.4985 & 0.3258 \\ 5.1320 & 15.0903 & 16.4037 & 7.7592 & 2.5989 \\ 10.5820 & 7.4392 & 17.1605 & 22.5170 & 19.0499 \end{bmatrix}$$

where μ_{jd} and σ_{jd} are mean and standard deviation matrices of emission probability density functions, rows indicate spatial states and columns indicate symbol dimensions. Spatial state 0 (the first row) is a "no data" state. Theoretically, the mean and standard deviation of every dimension of state 0 (the first row of both matrices) should be zero. The non-zero values arise from the fact that some observations are randomly clustered into the state during the unsupervised *k*-means training. For spatial state 1 to 4, each state has a significant peak dimension in the mean matrix (μ_{jd}), and this peak progressively migrates towards high dimensions. Since a high dimension corresponds to a high house price class, a high state number therefore tends to coincide with high price locations.

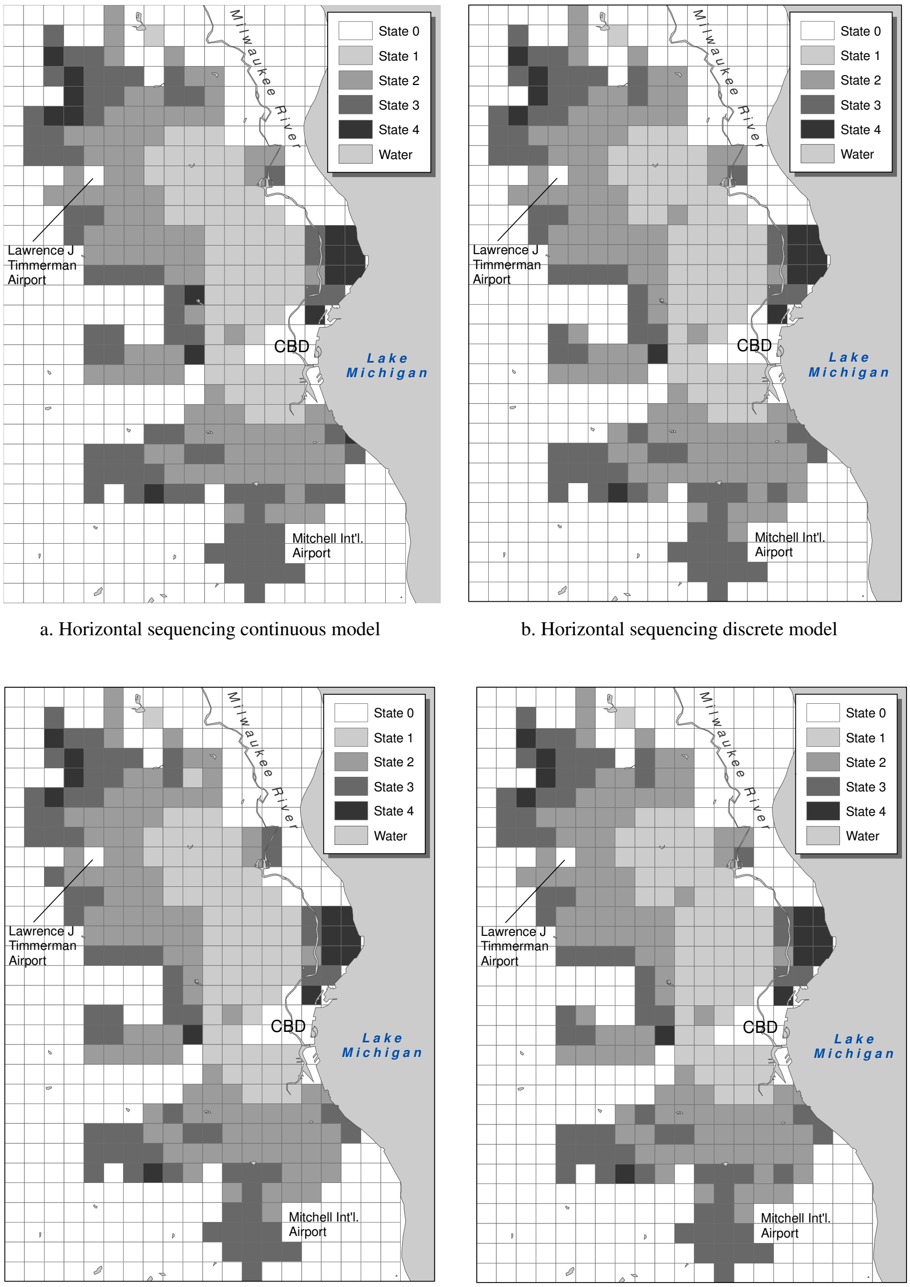
With discrete data handling, the *k*-means training algorithm computes emission probabilities for every spatial state to produce every unique observation symbol. Since 620 unique observation symbols are identified from the training dataset and five spatial states are discovered in this research, a total of 3100 (= 620 × 5) emission probability values are calculated. Because of space constraints in this paper, the unique symbols and corresponding emission probabilities are omitted.

5.2 Spatial States Discovered by the Viterbi Algorithm

Hidden spatial states discovered by the Viterbi algorithm are presented in Figure 4. The maps are results of different models based on various combinations of symbol sequencing and data continuity. Hidden spatial states are indications of the controlling effect of location on home prices. The four spatial states (1 to 4) agree well with the

Table 1. Discrepancies between HMM models (number of different cells)

		Horizontal sequencing		Vertical sequencing	
		Continuous	Discrete	Continuous	Discrete
Horizontal sequencing	Continuous				
	Discrete	10 (1.67%)			
Vertical sequencing	Continuous	11 (1.83%)	6 (1.00%)		
	Discrete	10 (1.67%)	0 (0.00%)	6 (1.00%)	



a. Horizontal sequencing continuous model

b. Horizontal sequencing discrete model

c. Vertical sequencing continuous model

d. Vertical sequencing discrete model

Fig. 4. Hidden spatial states identified by hidden Markov models

traditional urban zones shown in Figure 3, with spatial state 1 coinciding with the poor inner city sub-areas, state 2 with a transitional zone, state 3 with the affluent suburbs, and spatial state 4 with the extremely wealthy lakefront neighborhoods.

The great similarity in the resulting maps suggests stability in the Hidden Markov Model approach. Only by careful scrutiny may differences between the maps be distinguished. Table 1 discloses discrepancies between models by showing the number and percentage of cells differing in states. The table indicates that the overall difference between models is very small with an average inconsistence of 1.19%.

5.3 Hidden States and Location Values

To further discover the magnitude of the impact of spatial states on housing price, a traditional hedonic model is used, in which sixteen structural variables and the four spatial states derived from the horizontal sequencing discrete model are used for linear regression. Table 2 is a list of the variables considered.

Table 2. Variables used in the hedonic housing price model

Variables	Description
Dependent:	
Houseprice	House price, in US dollars.
Independent:	
Lot_area	Lot area, in square feet
Building_area	Building area, in square feet
Building_age	Building age, in years
Num_rooms	Number of rooms
Bedrooms	Number of bedrooms
Num_baths	Number of bathrooms
New_home	New home, dummy variable, 1 for house_age $\leq$ 1, 0 otherwise
Has_basement	Has basement, dummy variable
Has_attic	Has attic, dummy variable
Has_fileplace	Has fireplace(s), dummy variable
Has_airconditioning	Has air conditioning, dummy variable
Has_powderroom	Has powder room, dummy variable
One_story	One story building, dummy variable
Two+_stories	Two or more story building, dummy variable
Attached_garage	Has attached garage, dummy variable
Detached_garage	Has detached garage, dummy variable
STATE1	Dummy variable
STATE2	Dummy variable
STATE3	Dummy variable
STATE4	Dummy variable

The dataset provides 6039 valid cases for regression analysis. The overall regression is satisfactory with a coefficient of determination R^2 = 0.646 (p-value = 0.0000). Coefficients indicate that, based on a constant coefficient $98,727, spatial state 1 incurs a reduction of $35,910 in housing price, spatial state 3 contributes an increase of $25,970, and spatial state 4 contributes an increase of $142,577. Variable state 2 is removed in the regression since it does not significantly contribute to the variance of the house price given that constant. Standardized coefficients indicate that building area (β = 0.435), state 4 (β = 0.323) and state 1 (β = -0.239) are the three most influential variables. However, *t*-tests of the coefficients indicate that the number of building stories (one_story: *p*-value = 0.177, two+_stories: *p*-value = 0.133) and garages (attached_garage: *p*-value = 0.533, detached garage: *p*-value = 0.839) are not significant. Moreover, these four variables show a much higher correlation to building area and number of rooms than to the dependent variable.

After removing those less influential variables and rerunning the regression, the coefficient of determination R^2 (0.643) is not significantly changed. However, the most influential variables become building area ($\beta = 0.424$), state4 ($\beta = 0.407$), state3 ($\beta = 0.356$) and state 2 ($\beta = 0.275$). Variable "state1" is removed this time since the regression constant becomes $11,969. Spatial state 2 now contributes +$37,614 to the house price, spatial state 3 contributes +$63,041, and state4 contributes +$179,835.

Table 3. Regression coefficients of spatial states

	Regression 1		Regression 2		Average Difference
	Coefficient	Difference	Coefficient	Difference	
Constant	98,727		11,969		
State 1	-35,910		0		
State 2	0	35,910	37,614	37,614	36,762
State 3	25,970	25,970	63,041	25,427	25,699
State 4	142,577	116,607	179,835	116,794	116,701

Both regression models agree well in differences in coefficients of spatial states or the impacts of states. Table 3 is a summary of the spatial state coefficients and differences between them. The last column in the table shows the impacts of location, indicated by spatial states, on housing values. For example, the location represented by spatial state 4 is $116,701 more valuable than that of state 3.

6 Conclusion

We introduced the Hidden Markov Model and demonstrated that it can be an ideal approach for uncovering hidden spatial patterns. The HMM is a reliable approach to spatial pattern identification since it takes into account both variation of observed values and spatial dependence in pattern recognition. In other words, it captures both the local variability and regional trends – the geographic context. Spatial dependence is accounted for by transition probability and cross-class correlation rather than Euclidean distance so that the method is not restricted to any particular spatial pattern and can handle directional asymmetry. The HMM substantially differs from a traditional Markov chain in that it models Markov processes without known parameters. In addition, it is capable of working with data of any measurement scale and dimension.

The HMM was demonstrated in a case study that, using home sale price data, uncovered the urban spatial structure of the City of Milwaukee. House prices were assumed to be indicators of underlying spatial states and used to create symbols and observation sequences. These sequences then were used to build HMMs and uncover the hidden spatial states. House prices were not the target of this research nor was the model intended for predicting home prices. Therefore, temporal processes of home sale prices, important factors reflecting market evolution, were not included in the study.

Spatial data, especially random location point datasets, normally lack inherent sequences. To make HMM more sophisticated and versatile for geospatial problems, further research will be directed towards spatial data segmentation, symbolization and sequencing, Markov field definition, and data continuity handling.

References

1. Aycard, O., Mari, J.F., Washington, R.: Learning to automatically detect features for mobile robots using second-order Hidden Markov Models. International Journal of Advanced Robotic Systems 1(4), 233–250 (2004)
2. Basu, S.: Analysis of spatial autocorrelation in house prices. Journal of Real Estate Finance and Economics 17(1), 61–85 (1998)
3. Bierkens, M.F.P., Burrough, P.A.: The indicator approach to categorical soil data: I, Theory. Journal of Soil Science 44, 361–368 (1993)
4. Brunsdon, C., Fotheringham, A.S., Charlton, M.E.: Geographically weighted regression: a method for exploring spatial nonstationarity. Geographical Analysis 28, 281–298 (1996)
5. Can, A.: Specification and estimation of hedonic housing price models. Regional Science and Urban Economics 22, 453–474 (1992)
6. Cheshire, P., Sheppard, S.: On the price of land and the value of amenities. Economica 62, 247–267 (1995)
7. Cliff, A.D., Haggett, P.: On complex geographic space: Computing frameworks for spatial diffusion processes. In: Longley, P.A., Brooks, S.M., McDonnell, R., MacMillan, B. (eds.) Geocomputation: A Primer, pp. 231–256. Wiley, New York (1998)
8. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B 39, 1–38 (1997)
9. Dubin, R.A.: Spatial autocorrelation and neighborhood quality. Regional Science and Urban Economics 22, 433–452 (1992)
10. Elfeki, A., Dekking, M.: A Markov chain model for subsurface characterization: theory and applications. Mathematical Geology 33(5), 569–589 (2001)
11. Ester, M., Kriegel, H.P., Sander, J.: Algorithms and applications for spatial data mining. In: Miller, H.J., Han, J. (eds.) Geographic Data Mining and Knowledge Discovery, pp. 160–187. Taylor & Francis, London (2001)
12. Evans, A.W.: The economics of residential location. Macmillan, London (1973)
13. Fong, E., Shibuya, K.: The spatial separation of the poor in Canadian cities. Demography 37(4), 449–459 (2000)
14. Getis, A., Ord, J.: The analysis of spatial association by use of distance statistics. Geographical Analysis 24, 189–206 (1992)
15. Goovaerts, P.: Stochastic simulation of categorical variables using a classification algorithm and simulated annealing. Mathematical Geology 28, 909–921 (1996)
16. Grether, D.M., Mieszkowski, P.: The effects of nonresidential land uses on the prices of adjacent housing: some estimates of proximity effects. Journal of Urban Economics 8, 1–15 (1980)
17. Han, J., Kamber, M., Tung, A.K.H.: Spatial clustering methods in data mining. In: Miller, H.J., Han, J. (eds.) Geographic data mining and knowledge discovery. Taylor & Francis, New York (2001)

18. Hughes, J.P., Guttorp, P., Charles, S.P.: A non-homogeneous hidden Markov model for precipitation occurrence. Journal of the Royal Statistical Society (Series C): Applied Statistics 48(1), 15–30 (1999)
19. Iceland, J.: Beyond black and white: metropolitan residential segregation in multiethnic America. Social Science Research 33(2), 248–271 (2004)
20. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, Chichester (1990)
21. Levine, M.V.: Suburban sprawl and the "secession" of the affluent: metropolitan polarization in Milwaukee: 1987-1997. University of Wisconsin-Milwaukee Center for Economic Development Policy Research Report (1999), `http://www.uwm.edu/Dept/CED/publications/sprawl99.html`
22. Levine, M.V.: Metropolitan polarization in an era of affluence: income trends in metropolitan Milwaukee since 1990. University of Wisconsin-Milwaukee Center for Economic Development Policy Research Report (2002a), `http://www.uwm.edu/Dept/CED/publications.html`
23. Levine, M.V.: The economic state of Milwaukee.s inner city: 1970-2000, University of Wisconsin-Milwaukee Center for Economic Development Policy Research Report (2002b), `http://www.uwm.edu/Dept/CED/publications.html`
24. Levine, M. V. 2003. The two Milwaukees: separate and unequal, `http://www.uwm.edu/Dept/CED/publications.html`
25. Li, W.: Transiogram: A spatial relationship measure for categorical data. International Journal of Geographical Information Science 20(6), 693–699 (2006)
26. Li, W., Zhang, C.: A generalized Markov chain approach for conditional simulation of categorical variables from grid samples. Transactions in GIS 10(4), 651–669 (2006)
27. Lovell, B.C.: Hidden Markov models for spatio-temporal pattern recognition and image segmentation. In: International Conference on Advances in Pattern Recognition - eprint.uq.edu.au (2003)
28. Mari, J.F., Le Ber, F.: Temporal and spatial data mining with second-order hidden Markov Models. Soft Comput. 10, 406–414 (2006)
29. Miller, H.J.: Tobler's First Law and Spatial Analysis. Annals of the Association of American Geographers 94(2), 284–289 (2004)
30. Miller, H.J., Wentz, E.A.: Representation and Spatial Analysis in Geographic Information Systems. Annals of the Association of American Geographers 93(3), 574–594 (2003)
31. Milwaukee Neighborhood Identification Project, Milwaukee neighborhoods (2000) (Last time accessed August 17, 2007), `http://www.city.milwaukee.gov/displayFile.asp?docid=39&filename=/Public/map4.pdf`
32. Movellan, J.R.: Tutorial on hidden Markov models. Machine perception laboratory online tutorials (2003) (Last time accessed May 2, 2006), `http://mplab.ucsd.edu/tutorials/pdfs/hmm.pdf`
33. Nathan, K.S., Bellegarda, J.R., Nahamoo, D., Bellegarda, E.J.: On-line handwriting recognition using continuous parameter hiddenMarkov models. In: ICASSP 1993 (IEEE International Conference on Acoustics, Speech, and Signal Processing 1993), vol. 5, pp. 121–124 (1993)
34. Ord, J.K., Getis, A.: Local spatial autocorrelation statistics – distributional issues and an application. Geographical Analysis 27, 286–306 (1995)
35. Orford, S.: Valuing locational externalities: a GIS and multilevel modelling approach. Environment and Planning B: Planning and Design 29, 105–127 (2002)
36. O'Sullivan, D., Unwin, D.J.: Geographic information analysis. John Wiley and Sons, Inc., Hoboken (2003)

37. Palay, M.G.: 1973, Facts and Figures for Community Analysis. Milwaukee Urban Observatory, University of Wisconsin-Milwaukee, Milwaukee (1970)
38. Pamuk, A.: Geography of Immigrant Clusters in Global Cities: A Case Study of San Francisco, 2000. International Journal of Urban and Regional Research 28(2), 287–307 (2004)
39. Pena, J.M., Lozano, J.A., Larranaga, P.: An empirical comparison of four initialization methods for the K-Means algorithm. Pattern Recognition Letters 20, 1027–1040 (1999)
40. Rast, J.: Transportation equity and access to jobs in metropolitan Milwaukee. University of Wisconsin-Milwaukee Center for Economic Development Policy Research Report (2004), `http://www.uwm.edu/Dept/CED/publications.html`
41. Rabiner, L.R., Wilpon, J.G., Juang, B.H.: A Segmental K-means Training Procedure for Connected with Recognition Based on Whole Word Reference Patterns. AT&T Technical Journal 65(3), 21–31 (1986)
42. Rabiner, L.R.: A tutorial on hidden Markov model and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
43. Robertson, A.W.S.K., Smyth, P.: Hidden Markov models for modeling daily rainfall occurrence over Brazil. Technical Report UCI-ICS 03-27, Information and Computer Science, University of California, Irvine (2003)
44. Shekhar, S., Lu, C.T., Zhang, P.: A unified approach to detecting spatial outliers. GeoInformatica 7(2), 139–166 (2003)
45. Tanguay, D.O.: Hidden Markov models for gesture recognition. Thesis of Massachusetts Institute of Technology (1995)
46. Tjelmeland, H., Besag, J.: Markov random fields with higher-order interactions. Scandinavian Journal of Statistics 25, 415–433 (1998)
47. Tobler., W.: A computer movie simulating urban growth in the Detroit region. Economic Geography 46, 234–240 (1970)
48. Varel, O.J.: Examining Patterns of Neighborhood Change in the City of Milwaukee, 1980-1990: An Ecological Approach. University of Wisconsin-Milwaukee dissertation (1999)
49. Viovy, N., Saint, G.: Hidden Markov models applied to vegetation dynamics analysis using satellite remote sensing. IEEE Transactions on Geoscience and Remote Sensing 32(4), 906–917 (1994)
50. Wilkes, R., Iceland, J.: Hypersegregation in the twenty-first century. Demography 41(1), 23–36 (2004)
51. Wong, D.W.S., Lee, J.: Statistical Analysis of Geographic Information. John Wiley & Sons, Inc., New Jersey (2005)
52. Zhang, J., Goodchild, M.: Uncertainty in geographic information. Taylor and Francis, New York (2002)
53. Zhang, C., Li, W.: Markov chain modeling for multinomial land-cover classes. GIScience & Remote Sensing 42(1), 1–18 (2005)

Appendix A: Viterbi Algorithm

Problem: Given an HMM $\lambda = \{A, B, \pi\}$ and an observation sequence $O = \{O_1, O_2, \ldots O_T\}$, what is the most probable sequence of states underlying O?

First, the probability of producing the observation sequence O and an underlying state sequence S given an HMM can be derived by

$$P(O,S \mid \lambda) = P(O \mid S) \cdot P(S \mid \lambda) = \pi_1 b_1(O_1) a_{1,2} b_2(O_2) \ldots a_{T-1,T} b_T(O_T) \tag{10}$$

Define a cost function U as

$$U(s_1, s_2, \ldots, s_T) = -[\ln(\pi_1 b_1(O_1)) + \sum_{t=2}^{N} \ln(a_{t-1,t} b_t(O_t))] \tag{11}$$

Thus

$$P(O,S \mid \lambda) = e^{-U(s_1, s_2, \ldots, s_T)} \tag{12}$$

Now, the problem of maximizing $P(O, S \mid \lambda)$ becomes

$$\min U(s_1, s_2, \ldots, s_T) \tag{13}$$

Let's define $\delta_t(i)$ as the accumulated cost at state S_i at time t, and $\psi_t(j)$ as a state immediately prior to state S_j at time t-1 via which the transition cost is the least, i.e. with the highest transition probability. The problem can be resolved by the following iterations:

1) Initialization, for $1 \le i \le N$

$$\delta_1(i) = -\ln(\pi_i) - \ln(b_i(O_1)) \tag{14}$$

$$\psi_1(i) = 0 \tag{15}$$

2) Recursion, for $1 \le t \le T$, $1 \le j \le N$

$$\delta_t(j) = \min_{1 \le i \le N} [\delta_{t-1}(i) - \ln(a_{ij})] - \ln(b_j(O_t)) \tag{16}$$

$$\psi_t(j) = \arg\min_{1 \le i \le N} [\delta_{t-1}(i) - \ln(a_{ij})] \tag{17}$$

3) Termination

$$p^* = \min_{1 \le i \le N} [\delta_T(i)] \tag{18}$$

$$q^*_T = \arg\min_{1 \le i \le N} [\delta_T(i)] \tag{19}$$

4) Tracing back the optimal state sequence, for $t = T$-1, T-2, …, 1

$$q^*_t = \psi_{t+1}(q^*_{t+1}) \tag{20}$$

Finally, the state-optimized probability is e^{-p^*}, and the optimized state sequence is

$$Q^* = \{q^*_1, q^*_2, \ldots, q^*_T\} \tag{21}$$

Modeling Herds and Their Evolvements from Trajectory Data

Yan Huang[1], Cai Chen[2], and Pinliang Dong[2]

[1] Department of Computer Science
University of North Texas, Denton, Texas, USA
huangyan@unt.edu
[2] Department of Geography
University of North Texas, Denton, Texas, USA
caichen@unt.edu, pdong@unt.edu

Abstract. A trajectory is the time-stamped path of a moving entity through space. Given a set of trajectories, this paper proposes new conceptual definitions for a spatio-temporal pattern named *Herd* and four types of herd evolvements: *expand, join, shrink,* and *leave* based on the definition of a related term *flock*. Herd evolvements are identified through measurements of *Precision, Recall,* and *F-score*. A graph-based representation, *Herd Interaction Graph*, or *Herding*, for herd evolvements is described and an algorithm to generate the graph is proposed and implemented in a Geographic Information System (GIS) environment. A data generator to simulate herd movements and their interactions is proposed and implemented as well. The results suggest that herds and their interactions can be effectively modeled through the proposed measurements and the herd interaction graph from trajectory data.

Keywords: Spatio-temporal Data Mining, Spatial Patterns, Spatial Evolvements, Herd Evolvements.

1 Introduction

Many moving objects exist in the air, on the ground, or in the ocean. The detection and description of spatio-temporal patterns are essential for better understanding of the behaviors of moving objects (animals, vehicles, and people). For example, models of movements can be used to study the ecology of animal behaviors, habitat preferences, and the dynamics of population densities [4]. Other application examples of the analysis of moving objects include studies in socio-economic geography [8], transport analysis [19], defense, and surveillance [18]. With increased accessibility to data collected by Global Positioning Systems (GPS), radio transmitters, and other location-aware devices, the processing, storage, management, mining and analysis of data, information and knowledge related to moving objects have been a research focus in the last few years.

Trajectory analysis has been the focus of many research efforts recently and is of particular interest in many fields. A trajectory is a sequence of time-stamped point locations describing the path of a moving object with identity e over a period of time. Given a set of trajectories of a set of entities, the problem that this paper deals with is to mine grouping dynamics of moving entities described by their trajectories over time. Assuming the time used by an entity between any two consecutive locations is the same (uniformed sampling), then a trajectory can be represented by $tj(e, \tau) =< p_1, p_2, \ldots, p_\tau >$, where τ is the length of the trajectory and $tj(e, \tau)[t]$ represents the point locations visited by the entity e at time snapshot $t (1 \leq t \leq \tau)$.

Recently, algorithms to find *flocks* have been proposed to identify groups of entities that travel together for an extended period of time [10]. Formally, given the trajectories of a set of n entities, a time interval I of at least k consecutive snapshots, and a distance r, a flock f(m, k, r) is a set of at least m entities such that for every snapshot t in time interval I, there is a disk of radius r that contains all the m entities [10].

The concept of *flock* is based on several parameters to be provided by a user: the minimum number of entities m, the duration of a time interval k, and the radius r. The focus of *flock* is more on query-based data exploration. However, there are several limitations with a flock-based approach: (1) Entities, e.g. caribous, do not always gather in circular shapes. Finding at least m entities in a radius r, may not give an accurate picture of a *flock*, which most likely roams in arbitrary shapes; (2) Given m, k, and r, the flocks found will have many overlaps. For example, flock F_1 may consist of $m = 100$ entities E traveling within radius r for a time interval I of at least length $k = 1000$. For a query with $m = 50, k = 500$, any subset of E will be qualified as a flock and be returned as a result, which will lead to C_{100}^{50} flocks (choose 50 from 100 entities). Furthermore, for each flock in the C_{100}^{50} flocks, any sub-interval of the I will result in a flock, leading to $C_{100}^{50} \times (1000 - 500 + 1)$ flocks. Although algorithms for discovering longest flocks have been proposed [10], the problem of combinatorial explosion related to entities has not been addressed; (3) Flocks *move* and *evolve* over time. Because of these limitations, there is a need to discover how flocks interact with each other over time throughout the observations. In this paper we focus on modeling *group* traveling patterns and the evolvements and interaction of these groups.

The major contributions of this paper are: (1) A new concept *herd* is proposed for spatio-temporal patterns along with four types of spatial evolvements: *expand, join, shrink,* and *leave*; (2) Clustering-based methods are used to detect herd snapshots in trajectory data, and mathematical measurements of *Precision*, *Recall*, and *F-score* are proposed to identify herd evolvements; (3) A graph-based representation for herd evolvement is designed and implemented as an extension to a Geographic Information System (GIS). A herd evolvement simulator is implemented.

2 Related Work

Time is an essential dimension for analyzing and interpreting real-world evolution. In paper [6], the authors presented a set of design patterns for modeling spatio-temporal processes expressed in an object-relationship data model. They also claimed that describing geometric transformations of an independent spatial entity implies changes on four orthogonal attributes: shape (form of its boundary), size (area of its interior), orientation (compass direction of its major and minor axes), and location (position of its gravity center measured with geographic or Euclidean coordinates). In order to integrate phenomena that change over space and time in real-world phenomenon, a better understanding of the underlying components of change and how people reason about change are needed. In paper [12], the authors proposed a qualitative representation of changes. It offers a classification of changes based on object identity and the set of operations that either preserve or change identity. These operations can be applied to single or composite objects and combined to express the semantics of sequences of change. The authors also developed a visual language to represent the various types of change, and provided examples to illustrate the application of this language. Later, the authors used a temporal zooming approach to detect and navigate the spatio-temporal changes [11,13]. These approaches do not deal specifically with trajectory data.

Trajectory data analysis can be divided into two basic categories: single trajectory and multiple trajectory analysis. Since single trajectory data normally depicts one specific moving entity, single trajectory data analysis mainly focuses on looking for individual spatial patterns, and creating predictive models for the moving entity [2,20]. The predication models are useful for applications such as providing real time traffic information if the next trip stops can be predicted. In some applications, object movements obey periodic patterns or follow similar routes over regular time intervals. Effective data mining algorithms have been proposed to find spatio-temporal periodic patterns [5,17].

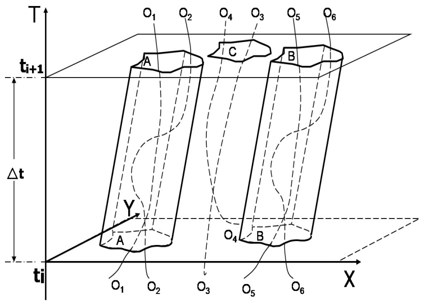
In recent years, there has been increased interest in analyzing spatial-temporal patterns and moving paths of wild animals [1] using multiple-trajectory data analysis. Geographic data mining approaches have been proposed to detect generic spatial-temporal patterns such as flock, leadership, convergence, and encounter in geospatial data [15,3,16]. A method to calculate the longest duration of flocks and meetings in spatial-temporal data has been proposed [10]. Definition of moving clusters and efficient algorithms to identify them have been proposed [14]. A recent work tries to find a set of individual trajectories that share the property of visiting the same sequence of places with similar travel times [9], which is related but not the focus of this paper.

Our work is related to and goes beyond finding moving clusters or identifying flocks. We provide a clustering-based definition of *herds* and further model *quantitative* changes and *qualitative* changes of the herds and their interactions.

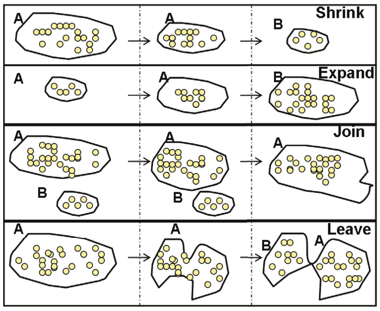
Our work is also related to spatial clustering. For static datasets, clustering analysis can be either used as a stand-alone tool to get insight into the distribution of a dataset, to identify areas for further analysis, or as a preprocessing step for other algorithms operating on detected clusters. Clustering algorithms can be classified into partition based, hierarchical clustering, density based, and model based methods. A density based clustering algorithm, e.g. DBSCAN [7], is attractive when one needs to identify arbitrary shaped clusters. In some applications, the location and content of spatial clusters may change over time, which requires a formal definition for moving clusters and algorithms for automatic discoveries [14].

3 Herd Evolvements

In this section, we introduce the ideas of herd and herd evolvements through visual illustrations and formal definitions. Figure 1(a) illustrates six trajectories $(O_1, O_2, .., O_6)$ over two time snapshots. From Figure 1(a), suppose we have A, B clusters in snapshot t_i and they will evolve into A, B, C clusters in snapshot t_{i+1}, where C is a new cluster formed in t_{i+1}. Please note that between the two time snapshots, the details of the trajectories are not perceivable due to the sampling granularity. Also, a cluster should include more entities than those illustrated in the figure.



(a) Trajectories and Herds



(b) Illustration of Herd Evolvements

Fig. 1. The Spatial Evolvements

Thus, if we take a snapshot of the locations of n entities at time t, we can spatially identify proximate groups of entities. We call each group consisting of a set of entities at time t a *herd snapshot*. Several herd snapshots over time may be related to each other and represent the *movements* and *evolvements* of herds over time. The mobility of entities causes *herd movement* whereas the membership change of entities causes *herd evolvement*.

The groups in each snapshot can be identified through various ways and we propose to use clustering algorithms for this purpose. A spatial cluster is a group of entities gathered in spatial proximity. Various clustering algorithms can be used including partitioning based, hierarchical, model based, and density based. We argue that a density-based clustering is more suitable in this application due to its following properties: (1) the ability to construct non-spherical clusters of arbitrary shapes; (2) the robustness with respect to noise in the data; (3) the ability to discover an arbitrary number of clusters without the need of specifying the number of clusters to find.

Once the herd snapshot concept is established, we investigate how the membership changes of herd snapshots can result in the dynamics of formation and deformation of new herds. For example, when a herd snapshot $H(t)$ of 100 entities at time t is joined by a few other entities, e.g. 5 entities, in the next snapshot, do we still have the same herd H? What about $2,000$ entities joined $H(t)$? Do we still have the same herd H? To summarize, the research issue here is how to characterize the herd evolvements using membership changes.

Let $H(t)$ represent a herd snapshot at time t and the herd H itself was formed at sometime t_0 before t. H's changes in the subsequent time snapshots after t_0 can be classified into: *quantitative* and *qualitative* changes. In a *quantitative change*, members leave H and new members may join H in small quantities. However, the herd can still be "reasonably well" represented by the initial members of H when H was formed, i.e. $H(t_0)$. In a *qualitative change*, members of H change so much that $H(t_0)$ is not a "good" representative of the herd anymore. Of course, the question is how to precisely define "reasonably well" and "good". We present the intuitive meaning of the four categories of qualitative changes here and propose the formal definitions in section 4:

– Expand: A herd H formed at time $t-i$ *expands* into a new herd H' at time t if $H'(t)$ contains "many" of the members of $H(t-i)$ but also contains "substantial" new members.
– Join: A herd H_1 formed at time snapshot $t-i$ *joins* a herd H_2 formed at at time snapshot $t-j$ to form a new herd H at time t, if $H(t)$ is "similar" to the herd H_2 and contains majorities of the members from herd H_1.
– Shrink: A herd H formed at time $t-i$ *shrinks* into a new herd H' at time t if $H(t-i)$ contains "many" of the members of $H'(t)$ but also contains "substantial" other members not in $H'(t)$.
– Leave: A herd H_1 *leaves* a herd H formed at time snapshot $t-i$ at snapshot t if the majorities of members of $H_1(t)$ are also in $H(t-i)$. Further more, the herd formed by the remaining members, denoted by $H_2(t)$, is "similar" to the herd H.

Figure 1(b) illustrates the four kinds of evolvements in three snapshots. In the *shrink* case, herd A went through a *quantitative change* first and then a *qualitative change*, to shrink into a new herd B; in the *expand* case, herd A went through a *quantitative change* first and then a *qualitative change*, to expand into a new herd B; in the *join* case, herd B joined herd A to form a new herd with name A due to the dominance of A in the new herd (reasons for using label A for the new herd will be discussed in section 4); in the *leave* case, herd A went through a *quantitative change* first and then herd B leaves herd A and the remaining herd is still labelled A (reasons for labeling one of the new herds as A will be cleared after section 4).

4 Measurements of Herd Evolvements

The *Precision(P), Recall(R) and F-Score(F)* measurements have been traditionally used for evaluating the performance of information retrieval systems. For a query Q and the collection of documents retrieved by Q, the measures assume a ground truth notion of relevancy: every document is known to be either relevant or non-relevant to Q. Intuitively, *recall* is the fraction of the documents that are relevant to the query and are successfully retrieved; and *precision* is the fraction of the documents retrieved that are relevant to what the user is querying for. The formal definitions are presented as follows:

$$Recall = \frac{|relevant\ documents| \bigcap |retrieved\ documents|}{|relevant\ documents|}$$

$$Precision = \frac{|relevant\ documents| \bigcap |retrieved\ documents|}{|retrieved\ documents|}$$

With the definitions of precision and recall, we can expound the meaning of F measure: the weighted harmonic mean of precision and recall. The formal definition of the F-score or balanced F-score is:

$$F = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

If we treat moving entities in trajectory data as documents in retrieval system, we can adopt the above measurements to the areas of spatio-temporal analysis. Thus, let $H(t)$ be a herd snapshot at time t, i.e. the members of herd H at time t, and let $H'(t+i)$ be another herd snapshot at time $t+i$, we adopt the *precision (P), recall (R)*, and *F-score (F)* measurements to model the relationship between $H(t)$ and $H'(t+i)$ as follows:

$$R(H(t), H'(t+i)) = \frac{|H(t) \bigcap H'(t+i)|}{|H(t)|}$$

Intuitively, $R(H(t), H'(t+i))$ measures the percentage of $H(t)$ that continue to exist in $H'(t+i)$. Thus, the more the entities left between t and $t+i$ from $H(t)$, the lower the value of $R(H(t), H'(t+i))$ is.

$$P(H(t), H'(t+i)) = \frac{|H(t) \bigcap H'(t+i)|}{|H'(t+i)|}$$

Intuitively, $P(H(t), H'(t+i))$ measures the percentage of $H'(t+i)$ that come from $H(t)$. Thus, the more the new entities joined in $H(t)$ between t and $t+i$, the lower the value of $P(H(t), H'(t+i))$ is.

$$F(H(t), H'(t+i)) = \frac{2 \times P(H(t), H'(t+i)) \times R(H(t), H'(t+i))}{P(H(t), H'(t+i)) + R(H(t), H'(t+i))}$$

$F(H(t), H'(t+i))$ represents the combined results of members left and new members joined and it ranges from 0 to 1, where 0 indicates $H(t)$ is completely different from $H'(t+i)$ and 1 indicates $H(t)$ and $H'(t+i)$ consist of exactly the same members.

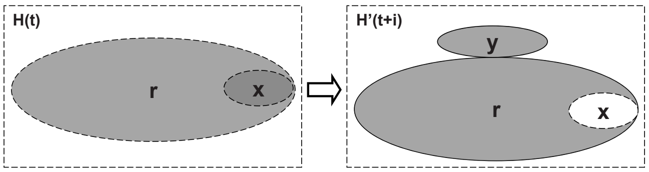


Fig. 2. Generic Herd's Evolvements from t to $t+i$ (all symbols, e.g. r, denote the areas delineated by their closest boundaries)

4.1 Measuring Generic Herd Evolvements Using $\boldsymbol{R}, \boldsymbol{P}, \boldsymbol{F}$

In this section, we describe the scenario and criteria to precisely define evolvements of herds.

In Figure 2, let the $|r \cup x|$ represent the members of a herd snapshot $H(t)$ at time t. At time $t+i$, suppose we detect another herd snapshot $H'(t+i)$ consisting of $r\cup$, where x is the set of entities that escaped from $H(t)$ between t and $t+i$ time snapshots, and y is the set of entities that joined in $H(t)$ between t and $t+i$ time snapshots, then the P, R, and F can be formulated using r, x and y. The *recall* is:

$$R(H(t), H'(t+i)) = \frac{|r|}{|r| + |x|}$$

and the *precision* is:

$$P(H(t), H'(t+i)) = \frac{|r|}{|r| + |y|}$$

And *F-score* will be(please note that r,x,y are disjoint):

$$F(H(t), H'(t+i)) = \frac{2 \times P \times R}{P + R} = \frac{2 \times \frac{|r|}{|r \cup y|} \frac{|r|}{|r \cup x|}}{\frac{|r|}{|r \cup y|} + \frac{|r|}{|r \cup x|}} = \frac{2 \times |r|}{2 \times |r| + |x| + |y|}$$

Thus, we propose the following criteria to define the *quantitative* and *qualitative* evolvements of herd:

1. When the sum of the number of the escaped members (i.e. x) and the number of the newly joined members (i.e. y) is less than the number of the remaining members of $H(t)$ in $H'(t+i)$ (i.e. r), we think that conceptually the herd can still be reasonably represented by the original members of H and claim that $H(t)$ underwent a *quantitative change* to $H'(t+i)$ and $H'(t+i)$ is the same herd as $H(t)$. That is, given $|x| + |y| < |r|$ or equivalently when:

$$F(H(t), H'(t+i)) > \frac{2}{3}$$

the change is *quantitative*;
2. When the sum of the number of the escaped members (i.e. x) and the number of the newly joined members (i.e. y) is no less than the number of the

remaining members of $H(t)$ in $H'(t+i)$(i.e. r), conceptually, we think the herd is NO longer the same herd and has undergone a *qualitative change*. We say that $H(t)$ underwent a *qualitative change* to $H'(t+i)$ and $H'(t+i)$ is NOT the same herd as $H(t)$.

That is, given $|x| + |y| \geq |r|$, or equivalently when:

$$F(H(t), H'(t+i)) \leq \frac{2}{3}$$

the change is *qualitative*. This case can be further divided into the following scenarios:

(a) When $|x| \geq |r|$ and $|y| < |r|$, or equivalently,

$$R(H(t), H'(t+i)) = \frac{|r|}{|r \cup x|} \leq \frac{1}{2}, P(H(t), H'(t+i)) = \frac{|r|}{|r \cup y|} > \frac{1}{2}$$

we say that $H(t)$ shrank or left by others into $H'(t+i)$. In this case, the escaped members (i.e. x) outweigh the remaining members of $H(t)$ in $H'(t+i)$ (i.e. r) and the remaining members of $H(t)$ in $H'(t+i)$ (i.e. r) outweigh the newly joined members (i.e. y).

(b) When $|y| \geq |r|$ and $|x| < |r|$, or equivalently,

$$R(H(t), H'(t+i)) = \frac{|r|}{|r \cup x|} > \frac{1}{2}, P(H(t), H'(t+i)) = \frac{|r|}{|r \cup y|} \leq \frac{1}{2}$$

we say that $H(t)$ expanded or be joined into $H'(t+i)$. In this case, the newly joined members (i.e. y) outweigh the remaining members of $H(t)$ in $H'(t+i)$ (i.e. r) and the remaining members of $H(t)$ in $H'(t+i)$(i.e. r) outweigh the escaped members (i.e. x).

(c) Otherwise, $H(t)$ and $H'(t+i)$ do not have a relationship. If $H(t)$ does not find any other herd snapshot at $t+i$ related to it, H simply disappears at time $t+i$.

Please note that the threshold of $\frac{2}{3}$ for the *F-score* and the threshold of $\frac{1}{2}$ for the precision and recall are decided by the "majority rules" and may be modified based on biological rules to fit specific application domains.

We summarize the conditions of various *qualitative changes* in Table 1.

Table 1. Qualitative Changes Based on P, R When $F \leq \frac{2}{3}$

$F < \frac{2}{3}$	$P > \frac{1}{2}$	$P \leq \frac{1}{2}$
$R \leq \frac{1}{2}$	Shrink or Split	No Relationship
$R > \frac{1}{2}$	No Relationship	Expand or Merge

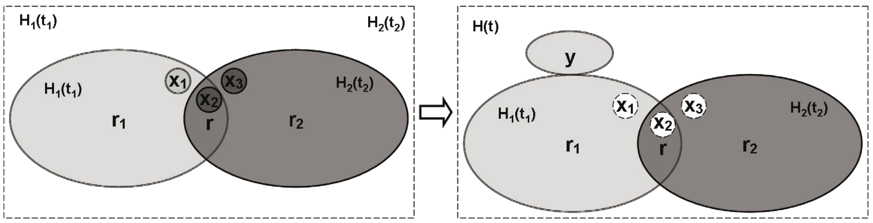


Fig. 3. Herds Formed at t_1 and t_2 Merge at t(all symbols, e.g. r, denote the areas delineated by their closest boundaries)

4.2 An Additional Concern in Labeling New Herds

If two or more herd snapshots are merged into a larger one at time $t+i$, will more than one of the merging herds claim that the newly formed herd is the same herd as themselves (both changes are not *qualitative*)? If this happens, we have an identity problem where we do not know how to call the newly formed herd.

In Figure 3, assume we have two herds H_1 and H_2 formed at t_1 and t_2 respectively, where $H_1(t_1)$ contains $r_1 \cup r \cup x_1 \cup x_2$ and $H_2(t_2)$ contains $r_2 \cup r \cup x_2 \cup x_3$. The two herds merged in the snapshot t into $H(t)$. At the same time $x_1 \cup x_2 \cup x_3$ escaped and did not participate in $H(t)$. But y joined at the same time. So at the end, the member set of $H(t)$ consists of $r_1 \cup r_2 \cup r \cup y$. To capture the inheritances among herds, we should name *Herd H(t)* either H_1 or H_2 if one of *F-scores* is larger than $\frac{2}{3}$. If both herds $H_1(t_1)$ and $H_2(t_2)$ went through *quantitative changes* and became $H(t)$, i.e. $H(t)$ is both $H_1(t_1)$ and $H_2(t_2)$, we have problems in labeling the descendant herds (e.g. *H(t)* in this case). Here, based on conceptual definition of *quantitative change*, if both herds went through *quantitative changes*, we have:

$$|x_1 \cup x_2| + |r_2 \cup y| < |r_1 \cup r|$$
$$|x_2 \cup x_3| + |r_1 \cup y| < |r_2 \cup r|$$

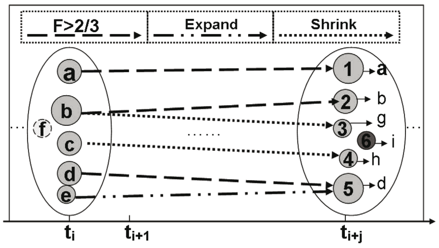
In fact, it is possible for the two equations above to hold. For example, when x_1, x_2, x_3, y are empty, $r_1 = r_2$, and r is not empty, it is obvious both equations hold.

In order to preserve herds evolvements, we propose a ranking strategy to establish the inheritance relationship among *Herds* at different snapshots. Specifically, if a herd at the current time snapshot has *F-scores* greater than $\frac{2}{3}$ with several herds formed previously, we rank the herds according to the *F-scores* and chose the one with the highest *F-score* as the label for the current herd.

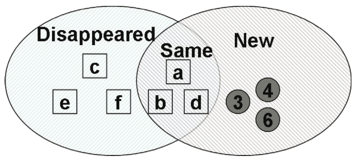
5 Herd Interaction Graph

Now we are ready to introduce the *Herd Interaction Graph* (or *Herding*) using various herd evolvements defined in the previous sections. For a given starting time t_{start} that we start to observe the herds, we find the *core member set* CM of a herd H. The core member set CM defines and represents the herd H until the actual member set of H deviates qualitatively from CM. Then H disappears and a new herd may emerge with a new CM'. Formally, we define the concept of a *core member set* as follows:

Definition 1 (Core Member Set). Given the trajectories of a set of n entities of length τ and a clustering algorithm to cluster the locations of the entities at each time snapshot: (1) When $t = t_{start}$, a cluster CM defines and initiates a herd H and is called the core member set of the herd H and can be denoted by $H(t)$. H will continue to exist and be represented by $H(t)$ until H's actual member set $H(t+i)$ at time $t+i$ is *qualitatively* different from $H(t)$ (where *F-score* $\leq \frac{2}{3}$); (2) When $t \neq t_{start}$, a cluster CM defines and initiates a herd H and is called the core member set of the herd H and can be denoted by $H(t)$ if and only if: CM is *qualitatively* different from any existing *core member set* formed in the previous time snapshots (where *F-score* $\leq \frac{2}{3}$); (3) A herd H together with its core member set CM disappears at t when there is no clusters found at t is a quantitative change of CM.



(a) Label Herds in snapshot t_{i+j}



(b) Update the *Core Member set*

Fig. 4. *Core Member Set* in Herds

In Figure 4(a), at the beginning time snapshot t_i, we assume there are 6 herds each represented by its core member set. They are labeled as a, b, c, d, e, f. After j time snapshots at snapshot t_{i+j}, we found 6 clusters labeled as 1 to 6 respectively. We need to see if these clusters are just some quantitative change of the earlier herds represented by their core member set or newly formed herds using R, P and *F-score*. We calculate their relationships with the core member set of a, b, c, d, e, f using R, P, and *F-score*. It turned out that $1, 2, 5$ are quantitative changes of a, b, d respectively, thus the herds a, b, d continue to exist and are still represented by their core member set at t_i. Furthermore, because cluster 5 is an expansion of cluster e (cluster 5 is also herd d), cluster e joined cluster 5. In a similar vein, cluster 3 (relabeled as g) is a shrinking of cluster b (cluster 2 is also herd b), so herd 3 left herd b. In addition, herd f disappeared, herd 4 (relabeled as h) is a shrinking of herd c, a new herd 6 (relabeled as i) appears. Figure 4(b) shows that in two snapshots, some herds disappear, some are new and the core member set of the common herds, e.g. b, d, will not change (although there are lost members and newly joined members).

5.1 Basic Algorithm to Generate Herd Interaction Graph

We are now ready to introduce our algorithm to generate the *Herd Interaction Graph* by determining the *quantitative changes* and *qualitative changes* in *Herds' Evolvements*.

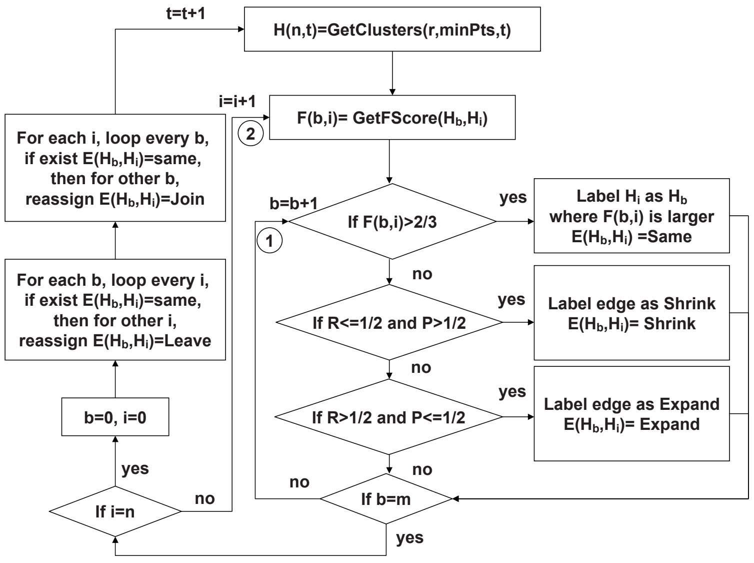

t=t+1
H(n,t)=GetClusters(r,minPts,t)
i=i+1
F(b,i)= GetFScore(H_b,H_i)
2
For each i, loop every b, if exist E(H_b,H_i)=same, then for other b, reassign E(H_b,H_i)=Join
b=b+1
If F(b,i)>2/3
yes
Label H_i as H_b where F(b,i) is larger E(H_b,H_i) =Same
1
no
For each b, loop every i, if exist E(H_b,H_i)=same, then for other i, reassign E(H_b,H_i)=Leave
If R<=1/2 and P>1/2
yes
Label edge as Shrink E(H_b,H_i)= Shrink
no
b=0, i=0
If R>1/2 and P<=1/2
yes
Label edge as Expand E(H_b,H_i)= Expand
yes
no
no
If i=n
no
If b=m
yes

Fig. 5. Flow Chart of the Algorithm to Generate Herd Interaction Graph

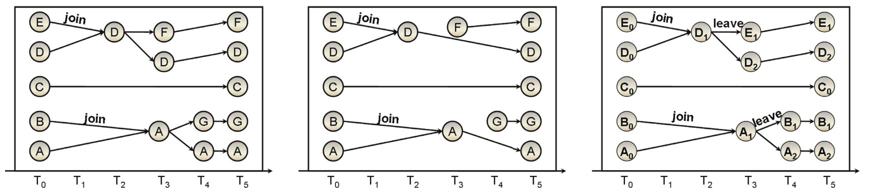
In Figure 5, we first use procedure *GetClusters(r, minPts, t)* to find clusters from the trajectory data at snapshot t via a clustering algorithm, e.g. DBSCAN, with two parameters: radius (r) and minimum points (minPts), and return the results in the array $H(n, t)$ where n is the number of clusters found. Here we assume we have m herds B formed previously represented by their *Core Member Sets*. To decide if the change from a cluster H_i and a core member set H_b is *quantitative* or *qualitative*, procedure *GetFScore(H_b,H_i)* calculates the *F-score* between them. We use $E(H_b, H_i)$ to represent the relationship, e.g. *expand*, between herd H_b and a cluster H_i.

In case of a *quantitative change*($F \geq \frac{2}{3}$), we rank the *F-scores*, label H_i as H_b, and assign $E(H_b, H_i) = same$, where $F(b, i)$ is the largest for all $b \in B$ if multi-ancestors are found. In case of a *qualitative change*, based upon the criteria we have discussed, we can determine that 1) herd H_b shrinks into herd H_i formed in this snapshot if $P > 1/2$ and $R \leq 1/2$; we then tentatively label the edge between H_b and H_i as *shrink*($E(H_b, H_i) = shrink$); 2) herd H_b expands into herd H_i formed in this snapshot if $P \leq 1/2$ and $R > 1/2$; We tentatively label the edge between H_b and H_i as *expand*($E(H_b, H_i) = expand$). After both loop 1 and loop 2 are over, we begin starting loop for b, i again and check if we need to reassign edges with *join* or *leave* by using the numbers of incoming and outgoing edges of herds.

Furthermore, for each b, we loop every i, if there is no H_i as the quantitative change of H_b, i.e.. $E(H_b, H_i) = \textit{same}$, we can determine that $H(b)$ disappeared. Similarly, for each i, we loop every b, if there is no H_b as the quantitative change of H_i, i.e. $E(H_b, H_i) = \textit{same}$, we can also determine that $H(i)$ is newly formed.

5.2 Intuitive Visualization of the Herd Interaction Graph

We are now able to produce the interaction graph (see Figure 6(a)) based upon the scenarios and algorithm we have proposed so far. We pay close attention to the intuitive meaning of the herd interaction graph to a user.



(a) Original Graph (b) Switch CM (c) Graph with Sub-herds

Fig. 6. Herd Interaction Graph

In Figure 6(a), for example, with herd E joining D, D will have to appear at snapshot T_2 although it does not have a qualitative change. Because the core member set of a herd does NOT change unless the herd discontinues to exist, new herds, e.g. F, will continue to compare with $D(T_0)$ which is the core member set of D rather than with $D(T_2)$. However, when F either left or shrank from $D(T_0)$, we need to draw a line from D at T_2 to F at T_3 as shown in Figure 6(b) which can be easily interpret as F left (or shrank) from $D(T_2)$ (D at T_2) instead of $D(T_0)$ (D at time T_0). As a result, there is a discrepancy between the actual meaning of a node in the graph and the intuitive meaning a user may observe.

To represent their interactions, we have to change the mechanism of producing interaction graphs by switching the core member set of a herd to *status-based core member set* of a herd whenever a herd label appears more than one time (see Figure 6(c)). Consequently, a herd will be labeled using the same herd name but with subscriptions. For example, in Figure 6(c), D appears in multiple snapshots, namely T_0, T_2, T_3 and T_5, and are labeled as D_0, D_1, and D_2 where the D_2 at T_5 is added to signify the ending of D.

Thus, in order to effectively symbolize and represent the evolvements and interaction of herds through the herd interaction graph, we introduce the concept of *sub-herd* to represent the core member set updates for more intuitive visualization.

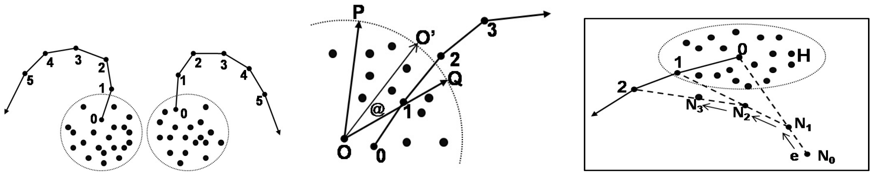
Definition 2 (Sub-herd(H)). Given a herd H formed at time t_0 whose core member is represented by $H(t_0)$, the sequential appearance of H on the herd interaction graph even without *qualitative change* are called *sub-herds* of H. Each sub-herd is represented by their members at the time when they appear on the graph (called *status-based core member set*) and is labeled as H_j where j is an increasing integer.

One more point that needs to be emphasized here is that, in Figure 6(b), a new herd G forms in snapshot T_4 calculated by using F, P, R, i.e. $F(G(T_4), A(T_0)) \leq \frac{2}{3}$. However, when we look backward to the herds formed in historical snapshots and determine which herd is the most similar to this new *herd* (the greater the F value is, the more similar the two *herds* are), we can detect that B in snapshot T_0 is actually not very different from G, and G should be labeled as B_1. In addition, using the core member set of A at T_3, we conclude that B_1 is a shrink of A_1. Combined with A_2, we decide B_1 left A_1 as shown in Figure 6(c). The situation for F in Figure 6(b) is similar. In addition, if *herd A(t)* disappears in next snapshot *t+1*, this herd is forced to appear on graph to highlight its completeness. For example E_1 disappears at time T_5 in Figure 6(c).

Overall in Figure 6(c), there are five herds $A, B, C, DandE$ at the first snapshot. Then herds B joined B at snapshot T_3 and then split into two herds similar to the original herds A and B at T_4; herd C stays the same throughout the entire observation; The behaviors of D and E are similar to those of A and B.

6 Algorithm Validations

We describe a herd simulator to facilitate the validation of the proposed concepts and algorithms for detecting herd and their interactions in this section. We also discuss the algebraic cost of the algorithms. Both the simulator and herd graph generation algorithms are implemented as an ArcGIS 9.x extension and source codes and details are available at http://groucho.csci.unt.edu/herding/.

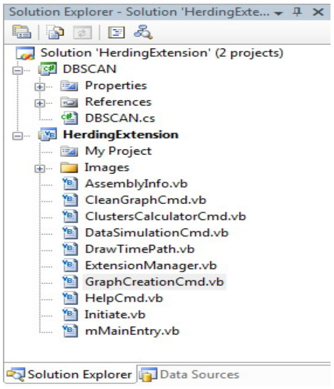


(a) Path (b) Spreading Angle (c) Joining a Herd

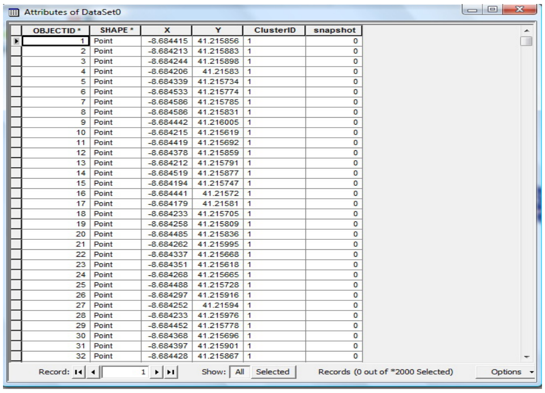
Fig. 7. Simulating Herds

6.1 Simulating Herds and Their Interactions

We have developed a herd simulator using VB.net to allow users to pre-define the moving pattern for each herd through a mouse. A set of clusters will be initially generated at time t_{start}. Then, as shown in Figure 7(a), a user can specify the path for each cluster (herd) where each point represents a snapshot (from 0 to 5 in the graph). Furthermore, a user can specify a spreading angle of a herd so that the core member of a herd will move randomly along the specified path and within the spreading angle. For example, in Figure 7(b), the moving entity O can randomly move to the territory within the area of PQO in the next snapshot. Generally, the greater the spreading angle, the more likelihood for the herd to split and disappear.



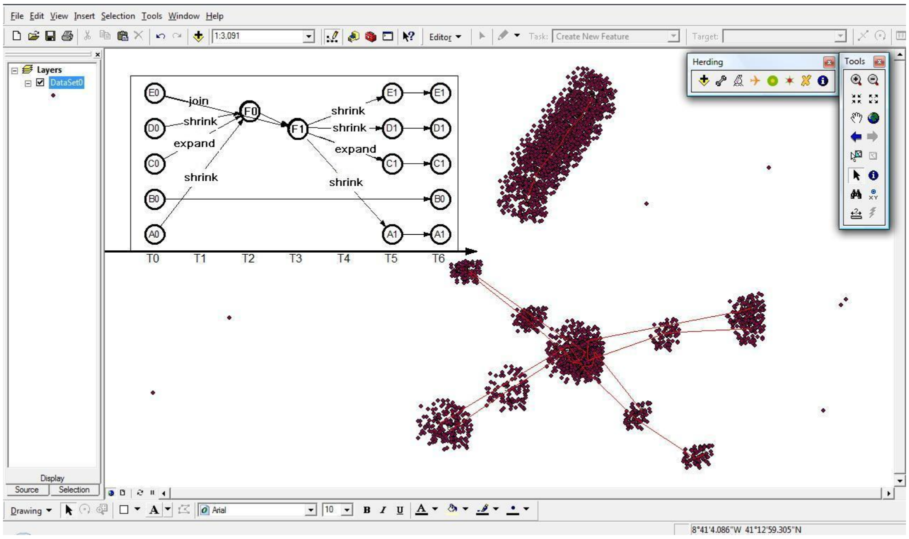
(a) Working IDE


Attributes of DataSet0

OBJECTID *	SHAPE *	X	Y	ClusterID	snapshot
1	Point	-8.684415	41.215856	1	0
2	Point	-8.684213	41.215883	1	0
3	Point	-8.684244	41.215898	1	0
4	Point	-8.684206	41.21583	1	0
5	Point	-8.684339	41.215734	1	0
6	Point	-8.684533	41.215774	1	0
7	Point	-8.684586	41.215785	1	0
8	Point	-8.684586	41.215831	1	0
9	Point	-8.684442	41.216005	1	0
10	Point	-8.684215	41.215619	1	0
11	Point	-8.684419	41.215692	1	0
12	Point	-8.684378	41.215859	1	0
13	Point	-8.684212	41.215791	1	0
14	Point	-8.684519	41.215877	1	0
15	Point	-8.684194	41.215747	1	0
16	Point	-8.684441	41.21572	1	0
17	Point	-8.684179	41.21581	1	0
18	Point	-8.684233	41.215705	1	0
19	Point	-8.684258	41.215809	1	0
20	Point	-8.684485	41.215836	1	0
21	Point	-8.684262	41.215995	1	0
22	Point	-8.684337	41.215668	1	0
23	Point	-8.684351	41.215618	1	0
24	Point	-8.684268	41.215665	1	0
25	Point	-8.684488	41.215728	1	0
26	Point	-8.684297	41.215916	1	0
27	Point	-8.684252	41.21594	1	0
28	Point	-8.684233	41.215976	1	0
29	Point	-8.684452	41.215778	1	0
30	Point	-8.684368	41.215696	1	0
31	Point	-8.684397	41.215901	1	0
32	Point	-8.684428	41.215867	1	0

Record: 1 Show: All Selected Records (0 out of *2000 Selected) Options

(b) Attribute Table in ArcGIS



(c) Generation of the Herd Interaction Graph

Fig. 8. Screenshots

By specifying the paths of herds to intersect, merge, and disappear as well as specifying the spreading speed of the herds, users can simulate the interaction of herds, and then apply our proposed algorithm to identify the herd interaction graph to verify their intended herd interactions.

To simulate the common effect of individuals joining an existing herd gradually over time, our simulator allows each entity not yet in a herd to randomly choose a herd and try to catch up with that herd over time by following the path of that herd using maximal speed. In Figure 7(c), the entity e chooses to follow herd H and always heads to the path of H over time, i.e. N_0 to N_1 to N_2 to N_3.

6.2 Generating the Herd Interaction Graph

The herd interaction graph representing the evolvements and movements of herds is then produced. Figure 6(c) is a screenshot generated by our tool, which is implemented as an ArcGIS 9.x extension. We used the Microsoft Visual Studio 2005 as our project IDE (see Figure 8(a)) and stored the trajectories into a geodatabase (See Figure 8(b)).

Figure 8(c) shows a few more screen shots of our extension to ArcGIS 9.x. In this run, we took a sample dataset with 500 moving entities stored as a geodatabase in ArcGIS and imported into ArcGIS desktop for further data manipulation.

6.3 Algebraic Cost Model of the Algorithm

We present the algebraic cost model of the proposed algorithm in this section and delay the optimization and performance evaluation to furture work due to the space constraint. Suppose we have N entities in our trajectory data, the running time of the clustering algorithm, i.e. *DBSCAN*, will be $O(NlogN)$. If we have T snapshots overall, the cost will be $O(T \times NlogN)$.

Let C represent the average number of clusters and k be the average number of core members for each cluster in each snapshot, then the cost of processing one snapshot will be $O(k^2 \times C^2)$ in order to determine the membership relationships of each pair of clusters using *Precision, Recall and F-score* measurements. Thus, the overall cost is: $O(T \times NlogN \times k^2 \times C^2)$.

7 Conclusion and Future Work

In this paper, 1) we introduced herds and their *quantitative* and *qualitative* changes during evolvements; 2) we defined *quantitative* and *qualitative* changes by leveraging measurements used in information retrieval, namely *Recall, Precision* and *F-score*; 3) we defined fours types of *qualitative* evolvements: *expand, join, shrink* and *leave*; 4) we introduced an effective method to identify these four types of spatial evolvements; 5) we developed a herd simulator to produce trajectories data in a Geographic Information System (GIS) environment; 6) we presented a graph-based representation - *Herd Interaction Graph* to represent herd interactions. The results suggest that herds and their interactions can be effectively modeled through the proposed measurements and the herd interaction graph from trajectory data. We also released the source code of the relevant implementations for public use. We plan to look into further optimization heuristics to improve the performance of the algorithm. We also plan to apply our algorithms to human mobility data to detect social events and their patterns.

References

1. Ankerst, M., Breunig, M.M., Kriegel, H.-P., Sander, J.: Optics: ordering points to identify the clustering structure. SIGMOD Rec. 28(2), 49–60 (1999)
2. Ashbrook, Starner, D.: Learning significant locations and predicting user movement with GPS. In: Sixth International Symposium on Wearable Computers, pp. 101–108 (2002)

3. Benkert, M., Gudmundsson, J., Hübner, F., Wolle, T.: Reporting flock patterns. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 660–671. Springer, Heidelberg (2006)
4. Brillinger, D.R., Preisler, H.K., Ager, A.A., Kie, J.G.: An exploratory data analysis (eda) of the paths of moving animals. Journal of Statistical Planning and Inference 122, 43–63 (2004)
5. Cao, H., Mamoulis, N., Cheung, D.W.: Mining frequent spatio-temporal sequential patterns. In: Proceedings of the Fifth IEEE International Conference on Data Mining, pp. 82–89 (2005)
6. Claramunt, C., Parent, C., Theriault, M.: Design Patterns for Spatio-temporal Processes. In: IFIP 2.6 Working Conference on Database Semantics,DS7 (1997)
7. Ester, M., Kriegel, H., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Second International Conference on Knowledge Discovery and Data Mining, pp. 226–231. AAAI Press, Menlo Park (1996)
8. Frank, A., Raper, J., Cheylan, J.-P.: Life and motion of spatial socio-economic units. In: Second International Conference on Knowledge Discovery and Data Mining. AAAI Press, Menlo Park (2001)
9. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: KDD 2007: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 330–339. ACM Press, New York (2007)
10. Gudmundsson, J., van Kreveld, M.: Computing longest duration flocks in trajectory data. In: GIS 2006: Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems, pp. 35–42. ACM Press, New York (2006)
11. Hornsby, K.: Temporal zooming. Transactions in GIS, 255–272 (2001)
12. Hornsby, K., Egenhofer, M.J.: Qualitative representation of change. In: Frank, A.U. (ed.) COSIT 1997. LNCS, vol. 1329, pp. 15–33. Springer, Heidelberg (1997)
13. Hornsby, K., Egenhofer, M.J.: Shifts in detail through temporal zooming. In: DEXA Workshop, pp. 487–491 (1999)
14. Kalnis, P., Mamoulis, N., Bakiras, S.: On discovering moving clusters in spatio-temporal data. In: International Symposium on Advances in Spatial and Temporal Databases (SSTD), pp. 364–381 (2005)
15. Laube, P., Imfeld, S.: Analyzing relative motion within groups of trackable moving point objects. In: 2nd International Conference on Geographic Information Science, pp. 132–144 (2002)
16. Laube, P., van Kreveld, M., Imfeld, S.: Finding remo detecting relative motion patterns in geospatial lifelines. In: Developments in Spatial Data Handling, pp. 201–215. Springer, Heidelberg (2005)
17. Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., Cheung, D.W.: Mining, indexing, and querying historical spatiotemporal data. In: KDD 2004: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 236–245. ACM Press, New York (2004)
18. Miller, J., Han, J.: Detecting outliers from large datasets. In: Geographic data mining and knowledge discovery, pp. 218–235. Taylor and Francis, Abington (2001)
19. Qu, Y., Wang, C., Wang, X.S.: Supporting fast search in time series for movement patterns in multiple scales. In: CIKM 1998: Proceedings of the seventh international conference on Information and knowledge management, pp. 251–258 (1998)
20. Sumpter, N., Bulpitt, A.J.: Learning spatio-temporal patterns for predicting object behaviour. Image Vision Comput 18(9), 697–704 (2000)

New Data Types and Operations to Support Geo-streams[⋆]

Yan Huang and Chengyang Zhang

University of North Texas
{huangyan,chengyang}@cs.unt.edu

Abstract. The volume of real-time streaming data produced by geo-referenced sensors and sensor networks is staggeringly large and growing rapidly. Queries on these geo-streams often require tracking spatio-temporal extent (e.g. evolving region) continuously in real time. The notion of real-time monitoring and notification requires support from a database capable of tracking and querying dynamic and transient spatio-temporal events as well as static spatial objects and sending out real-time notifications. In this paper, we leverage the work in data type based spatio-temporal databases and propose new data types called STREAM and their abstract semantics to support geo-stream applications. New operations on STREAM data types are defined and illustrated by embedding them into SQL.

1 Introduction

The advent of technologies in sensors and sensor networks is increasing human being's ability to interact with physical spaces in real time. Software tools that allow people to flexibly fuse, query, and make sense out of the data provided collectively by sensors in real time are very useful in many applications such as natural hazard monitoring, transportation, environmental modeling, and weather services.

Many sensors are geo-referenced and are excellent data sources for real-time situation monitoring and notification. The notion of real-time situation monitoring and notification requires support from a database capable of tracking and querying dynamic and transient spatio-temporal extents as well as static spatial objects and sending out real-time notifications. However, databases today lack this support.

In this paper, we deal with a special kind of data stream called geo-stream. A geo-stream refers to the moving and changing geometry of an object that is continuously fed into a geo-spatial data stream management system in real time. For example, the real time location of a moving object is a point geo-stream and the evolving spatial extent of a hurricane monitored in real time is a region

⋆ This work was partially supported by the Texas Advanced Research Program under Grant No. 003594-0010-2006 and by the National Science Foundation under Grant No. CNS-0709285.

geo-stream. We give motivating examples to illustrate the benefits of having a geo-stream database management system.

Motivation Example (Hazard Weather Notification System). Hazard weather warnings and watches such as flood watches and tornado warnings are continuously produced by National Weather Services and posted to its website using text messages as well as visual graphs to represent the spatial extents of the warnings. These geo-streams can be combined with a static spatial database to answer the following queries:

Q1. Notify me when my house is within 50 miles of the mandatory evacuation area of a forest fire.
Q2. Continuously list the addresses of all the houses traversed by flood in the past 2 days in Denton county.
Q3. Continuously list road segments that have been completely under floodwater for the past 24 hours.

Without a geo-stream database management system, sophisticated programming will be required to repeatedly issue queries to a spatial database and combine the results to answer the above questions. There are several problems with this approach: (1) it is difficult to decide how frequently the program should issue the queries to the spatial databases; (2) it is computationally expensive to find the latest data from all the historical data each time when the query is issued; (3) an integrated query optimization can not be performed by the database system and kept transparent from the user.

Modeling data streams collected by sensors in real time as a database system has been proven to be successful over the past years [5,19,4,3,2]. Several data stream management systems (DSMS) have been developed with some of them leading to startup companies [18]. Similar to traditional database systems, a DSMS system supports a declarative language which allows a user to express queries in a statement like fashion. These queries are processed and results are returned by the DSMS without exposing the details of the physical data organization to the user. A DSMS uses various query optimization techniques to ensure the quality of service.

However, current DSMS systems only include very rudimentary support for simple point locations [14]. Very little has been done to support moving and evolving spatial objects in REAL time such as trajectories and evolving regions. Spatio-temporal predicates such as traversed area and their implications to query optimization have not been investigated. Furthermore, most DSMS systems employ a simple relational table based paradigm and treat an individual tuple as the basic unit of operation. However, a data type based approach [8] where an entity's spatial extent over time can be managed as a data type and participate in common spatio-temporal predicates may be semantically simple. The paper makes the following contributions:

– We propose several new streaming data types to a spatio-temporal database system to support geo-streaming applications;
– We propose to represent window semantics using data types as well;

- We investigate the semantics of common spatio-temporal predicates and operations, and propose new ones meaningful on the proposed streaming data types;
- We illustrate the embedding of the new data types, operations, and predicates to a common database query language, i.e. SQL.

2 Related Work

Related work can be classified into four categories: spatio-temporal databases, moving object databases, streaming data management systems, and sensor network databases.

Our work is closely related to spatio-temporal databases [8,11,17,12,9] and to a data type based spatio-temporal database approach [8,11,12] in particular. This approach focuses on designing spatio-temporal data types, operations, and predicates and embedding them into a query language to support the storage and query of spatio-temporal data. The design goal of this approach is to find a set of data types and operations that are succinct, representative, consistent, and self-closed to support many applications in spatio-temporal domains. Spatial data types, e.g. points, lines, and polygons, their abstract semantics, and their discrete implementations in a computer system have been proposed. Spatial predicates and operations have been precisely defined. Together with non-spatial data types, e.g. string, spatial data types have been associated with time and elevated into temporal data types. Spatial predicates and operations have also been lifted. Spatio-temporal query languages that have SQL-like syntax have also been developed to embed the spatio-temporal data types and operations. However, to the best of our knowledge, supporting geo-streams in real time has not been addressed in this line of work. Instead, most assume a complete view of the spatio-temporal phenomena without considering their streaming nature. Prototype spatio-temporal database systems, such as Concert [17], Secondo [12], and Dedale [9], have been built with various levels of support for spatio-temporal data.

Moving object databases handle streaming location updates and support queries on them. They can be seen as streaming point data for *now* window in our proposed framework in most cases. There is a large body of work on indexing and query optimization on moving objects [13]. The indexing methods can be classified into two groups: disk based and memory based [16,15]. For volatile and dynamic objects, the maintenance cost of a disk based indexing could be prohibitive and memory based kinetic data structures are a better choice. Many query processing algorithms have been proposed for window queries and nearest neighbor queries. However, little has been done to support a suite of spatial streaming data types and their operations which is the focus of this work.

In a stream data management system, data arrives in the form of concurrent and continuous data streams. Queries on these data streams are typically continuous monitoring queries, involving both persistent relations and other time-varying streams, and emitting streaming data in real time as results. Data stream management systems have several significant characteristics different from those of traditional transactional and decision support systems. First, streams are generated on a regular, irregular, or bursty basis from many sources. A large number of streams from multiple sources and large volumes of data emitted from each source challenge traditional persistent relation and input/output reduction-based query processing paradigms. Second, stream-based applications require real-time response from the query processing system in order to trigger further actions despite bursty and unpredicted system loads. Third, queries on data streams involve not only data streams but also traditional relations. This requires an integrated model on both relations and data streams. Furthermore, queries on data streams are typically continuous queries, which require an intuitive, and semantically simple and clear query language/interface to specify queries and incorporate time window semantics. New processing paradigms and methods have been proposed and implemented in several stream processing systems [5,19,4,3,2] to achieve similar objectives. However, they can only handle streaming point locations naively [14] and do not have adequate support for evolving spatio-temporal extents. Efforts in developing data stream query languages ignore the support for dynamic and evolving geo-spatial objects from geo-streams.

Work in sensor (mote sized) databases [1,6,7] is focused more on reducing energy consumption and hiding the inherent heterogeneity and unreliability of sensor networks. Data input is modelled as relations instead of data types. In terms of spatio-temporal data and operation support, most sensors are static and location queries are limited to temporal range queries, e.g. find the average temperature in a given window. Some recent work addresses the in-network processing of spatial join of readings from sensors in spatial proximity [20]. We argue that treating individual and discrete sensor readings as the unit of operation is not sufficient. We need to support construction of spatio-temporal phenomena from raw sensor readings. For example, identifying soaked *regions* from distributed soil moisture sensors; and monitoring and tracking of the regions would be more useful than obtaining individual sensor readings. While abstracting spatio-temporal phenomena from raw sensor readings in real time is out of the scope of this paper and will be addressed in our future work, supporting data types such as moving regions as well as operations on them will be very useful.

In summary, current database systems DO NOT have a comprehensive set of spatial streaming data types and operations to support continuous queries involving both geo-streams and static geo-spatial objects.

Thus, the goal of this paper is to develop streaming data types, operations, predicates, and their language embeddings that allow a user to express continuous queries on both static and dynamic objects and monitor sensed phenomena in real-time. Towards this goal, we resolve two important research issues: (1) the definition of window semantics through a data type based approach; (2) the design of streaming data types, operations, and predicates under the proposed window semantics and the closure and consistency of the data types.

3 Streaming Data Types

For representing spatio-temporal objects and operations, many-typed algebra has been developed to ensure the closure of the operations among data types.

Because it also supports both static spatial and historical spatio-temporal data, a geo-stream database system should also support the basic data types available in these systems. We extend the work presented in [8,11,12] to represent the data types and operations. The data types marked by a $\circ$ at the end represent traditional data types in a non-streaming spatio-temporal database system:

	$\rightarrow$ BASE	*int, real, string, bool*	$\circ$
	$\rightarrow$ SPATIAL	*point, points, line, region*	$\circ$
	$\rightarrow$ TIME	*instant*	$\circ$
	$\rightarrow$ WINDOW	*now, unbounded, past*	*
BASE $\cup$ TIME	$\rightarrow$ RANGE	*range*	$\circ$
BASE $\cup$ SPATIAL	$\rightarrow$ TEMPORAL	*intime, moving*	$\circ$
(BASE $\cup$ SPATIAL $\cup$ RANGE) $\times$ WINDOW	$\rightarrow$ STREAM	*streaming*	*

In this notation, the abstract semantics of each data type is defined by its carrier set denoted by A_α where α is the data type. For example, the semantics of a point is represented by $A_{point} \equiv \{\mathbb{R}^2\} \cup \{\perp\}$ where $\perp$ represents the undefined value and the semantics of a point set is represented by $A_{points} \equiv \{P \subseteq \mathbb{R}^2 | P\, is\, finite\}$.

The detailed abstract semantics of BASE, SPATIAL, TIME, RANGE, and TEMPORAL data types have been defined elsewhere [8,11,12]. Briefly, the *line* data type is defined as a collection of simple curves (intersection of curves yields a finite number of intersection points). The *region* type is a collection of *faces* where a *face* is a set of points divided into interior, boundary, and exterior parts. The *instant* type is used to represent time and is isomorphic to the real numbers. Range is defined on BASE and TIME data types if a total order exists. A range has a pair of starting and ending values and includes all the values in between. The *intime* data type applied to a BASE or SPATIAL type α associates a time stamp with α. The *moving* data type applied to a BASE or SPATIAL data type α is a mapping from time to α. Please also note that all of the data types include an undefined value represented by $\perp$.

We assume that the abstract meaning of non-stream data types are properly defined and will focus on the new types namely WINDOW and STREAM in this paper.

3.1 WINDOW Types

In a data type based approach, the moving observation windows of a data stream are represented as data types. These data types can later be applied to BASE, SPATIAL, and RANGE data types to construct new streaming data types. We first define the abstract semantics of a WINDOW type that includes *now, unbounded*, and *past*.

At any time, the carrier of the type constructor *now* only has one value obtained from the system. Thus, $A_{now} \equiv$ current system time. We use *now* to refer to this value for simplicity.

At any time, an *unbounded* time window denotes the time interval starting from the system initial time until *now*. Thus, $A_{unbounded} \equiv (-\infty, now]$.

At any time, a *past* window specifies a time interval proceeding *now* and the semantics is defined by: $A_{past} \equiv \{X \subseteq A_{instant} | \forall x \in X(x \leq now) \wedge \forall y \in A_{instant}(x < y \leq now \Rightarrow y \in X)\}$.

3.2 STREAM Types

For a given BASE, SPATIAL, or RANGE type α and a WINDOW type ω, the type constructor *streaming* yields a mapping: $A_{streaming(\alpha,\omega)} \equiv \{f^c | f^c : A_{instant} \rightarrow A_{\alpha}$ is a partial function that is undefined for instants not in the window specified by ω and is continuously updated over time as the window changes according to the semantics of window type $\omega\}$.

For the *now* window and the *point* data type, $streaming(point, now)$ is a streaming pair consisting of a point location and the current time. A new pair replaces the old pair continuously over time. Thus, $A_{streaming(point,now)} \equiv \{f^c | f^c : A_{now} \rightarrow A_{point}$ where $\forall t \neq now, f^c$ is undefined $\}$.

For the *unbounded* window and the *point* data type, $streaming(point,$ $unbounded)$ is an ever increasing mapping from any time instant in the past to some point in space. Specifically, $A_{streaming(point,unbounded)} \equiv \{f^c | f^c : A_{instant} \rightarrow A_{point}$ where $\forall t \in (now, \infty), f^c$ is undefined $\}$.

For the *past* window and the *point* data type, $streaming(point, past)$ is a "streaming" mapping from any time instant in the time interval $(now - i, now]$ to some point in space where i is a time interval that can be optionally specified by the *past* data type. This is similar to the array data type in a programming language where the size of the array can be either specified at data type construction or left to the system. Let us assign the current system time *now* to a constant C at some time once. After time δ, the mapping for the interval $(C-i, C-i+\delta]$ becomes undefined and the mapping for $(C, C+\delta]$ is added (note that $C+\delta$ is the current value of *now*). Thus, $A_{streaming(point,past)} \equiv \{f^c | f^c : A_{instant} \rightarrow A_{point}$ where $\forall t \notin (now - i, now], f^c$ is undefined where i is a time interval$\}$.

We also assume if a window is not provided in the type constructor, the default is the *now* window. The abstract definitions for other streaming data types, e.g. *real* or *region*, can be defined in the similar manner. For all *streaming* types, we use the following naming convention: prefixing the BASIC or SPATIAL type with an *s* to represent its streaming version, e.g. *sint, sreal, sstring, sbool, spoint, spoints, sline, srange* and *sregion.*

4 Geo-stream Operations

Now the work boils down to the design of spatio-temporal operations and predicates under the window semantics. The desired properties of the operations are: meaningful so that they are useful, representative so that we do not have a cluttered predicate set, and general so that they can be used for many applications.

4.1 Windowing Operations on Streams

Streams can be formed from streams of different windows. For example, from a stream of an *unbounded* window, a stream of *now* can be formed. From a stream of *now*, a stream of the *past 2 hours* can also be formed although initially the two hours window is not complete. Thus, the operation **windowing:** $streaming(\alpha, \omega) \times \psi \rightarrow streaming(\alpha, \psi)$ maps one stream of window ω to a stream of the same data type α of another window ψ. To be consistent with window operations in general stream processing systems, we use the notion $streaming(\alpha, \omega)[\psi] \rightarrow streaming(\alpha, \psi)$ to represent the *windowing* operation in our language embedding later.

4.2 Projection to Domain and Range

Operations that apply on TEMPORAL data types and yield domain and range were provided in a spatio-temporal database. For example, the domain function **deftime**: $moving(\alpha) \rightarrow periods$ ($periods$ is a shorthand of the range of time $range(instant)$) returns the time intervals in which a function is defined. The **rangvalues**:$moving(\alpha) \rightarrow range(\alpha)$ operation returns values assumed over time as a set of intervals for a BASE type α in 1D space. Other operations in this category include *trajectory* which projects a moving object into lines in space and *traversed* which projects an evolving region into space and aggregates all projected regions into a larger region.

The projection to domain and range for streaming data types and functions have similar semantics except that the projection is continuously re-evaluated over the moving windows and the result is of streaming data types.

For example, the semantics of projection of a streaming region with window *unbounded* is to restrict the time to the window *unbounded* and apply the projection up-to *now*. This process is continuously applied as the window moves over time. In 1D space, the operation **rangevalues:** $streaming(\alpha, \omega) \rightarrow streaming(range(\alpha), now)$ for a BASE type α and a WINDOW type ω returns the values α assumed over streaming window ω for the current time *now*. For 2D space, streaming version of the operations such as *trajectory* and *traversed* can be defined. For example, the streaming operation **trajectory:** $streaming(\alpha, \omega) \rightarrow streaming(\alpha+, now)$ aggregates/elevates a set of streaming points of window ω to a high level object lines to represent a trajectory for the streaming window *now*.

Now suppose we are tracking caribous by satellite radio collars and the satellite collars turn on for some hours every day. The locations of the caribous are streamed back and modeled as caribou(name *string*, location *spoint*). Please be reminded that by default, the location of type *spoint* is of *now* window. And applying windowing operation on *location* is represented by $location[\psi]$ where ψ is a new window. The operation **deftime(caribou.location[unbounded]**) (note: *now* can be omitted since the default window of the location is *now*) returns the times when the caribou is tracked and the result is updated continuously. The operation **deftime(caribou.location[now])** returns the current time if the caribou is tracked now and $\perp$ otherwise. The operation **trajectory(caribou.location[past 2 hours])** returns the trajectories (represented by lines) of the locations of caribous projected to space in the past two hours continuously.

Table 1. Interaction of Streaming Values with Values in Domain and Range

Operations	Signature
atinstant	$streaming(\alpha, \omega) \times instant \rightarrow intime(\alpha)$
atperiods	$streaming(\alpha, \omega) \times periods \rightarrow moving(\alpha)$
present	$streaming(\alpha, \omega) \rightarrow streaming(bool, \omega)$
at	$streaming(\alpha, \omega) \times \alpha \rightarrow streaming(\alpha, \omega)$
at	$streaming(\alpha, \omega) \times range(\alpha) \rightarrow streaming(\alpha, \omega)$
passes	$streaming(\alpha, \omega) \times \beta \rightarrow streaming(bool, now)$

4.3 Intersection with Points and Point Sets in Domain and Range

Many queries need to use operations that relate the function values to their time domain or range. For TEMPORAL types, operations such as **initial** and **final** are useful to find the spatial extent at starting and ending times. For streaming applications, the starting and ending time of an event are difficult to determine. One solution is to ask users to explicitly associate each object of a streaming spatio-temporal extent with an **initialtime** and **finaltime**. However, these two attributes will not be part of the spatio-temporal or streaming data types and do not participate in spatio-temporal or streaming operations.

The **atinstant** and **atperiods** operation for TEMPORAL data type returns the spatial extent of the object for a given instant or a range of time. They may be useful for streaming data types if one wants to "intercept" the function values of a streaming data type at a specific time or time period. The result will be a constant data type, not a stream.

The **present** operation for a streaming data type α of window ω checks if the value of α exists during the time interval specified by the moving time window and returns a *sbool* data type of the same window. The **at** operation restricts the streaming data type's range to specific values. When the value of ranges is out of the given range, the operation returns an undefined value. For example, we can restrict the streaming real to the times when its values is between 0 and 1. The operation **passes** checks if the streaming value ever assumed the given values over the moving window. Table 4.3 summarizes the operations meaningful for streaming data types.

For the sample table of caribou, if we want to track for once where the caribous are for a given time t, we can use **atinstant(caribou.location[now], t)**. Once the time has passed t, this operation will not be useful anymore. The operation **present(caribou.location[past 3 days])** tells continuously at what times in the past three days the caribous are tracked. Let city(name *string*, location *point*) be a table storing the cities and their point locations. The operation **at(caribou.location[past 7 days],city.location)** returns continuously the caribous who had passed a city in the past 7 days.

4.4 Lifting Operations to Streaming Operations

For a spatial predicate or operation, its streaming version can be defined by converting one or more of the participating data types into its streaming version.

The semantics of such a straightforward conversion will need to be investigated. For example, the spatial predicate **intersect** with signature $region \times region \rightarrow bool$ can be first lifted into three streaming versions of **intersect** (we will discuss time shifted streaming operation shortly):

1. $streaming(region, \omega) \times region \rightarrow streaming(bool, \omega)$
2. $region \times streaming(region, \omega) \rightarrow streaming(bool, \omega)$
3. $streaming(region, \omega) \times streaming(region, \omega) \rightarrow streaming(bool, \omega)$

For **intersect** case 1, the predicate is to check if a streaming region intersects with a static region for each possible time instant in the valid moving window specified by ω and will produce a streaming *bool* as the result for the same moving window. Please note that the result is a streaming bool instead of a single TRUE or FALSE value. We further assume, when all the values in the given window ω is TRUE, the value of the result is TRUE. Similarily, the result is FALSE when all the values in the given window ω is FALSE. Otherwise, the result is unknown and further operations such as **at** or **duration** will be needed if a TRUE or FALSE is desired as the result which will be shown shortly using an example.

For **intersect** case 2, since **intersect** is a symmetric predicate, the semantics will be the same as that of case 1. For asymmetric operations and predicates, the semantics can be inferred straightforwardly from the static versions of the predicates. For **intersect** case 3, it returns a streaming bool of window ω representing the intersect results of two streaming regions of the same streaming window ω at each instant in the window. The lifting of other spatial predicates can be defined in the similar manner.

In addition, if we allow the time shifted (specified by an *instant* data type representing the time shift) **intersect**, the predicate can be more meaningful. For example, one may want to continuously see if the zones of high ozone intersect with the zones of high ozone for the same time last year. A time shifted **intersect** can be represented as:

$$streaming(region, \omega) \times moving(region) \times instant \rightarrow streaming(bool, \omega)$$

where the moving region represents a moving and evolving region (usually in the past). Similarily, operations between two streaming data types can be parameterized with a time shift as well.

However, instead of adding a time shift parameter to each operation, it would be more flexible and elegant to introduce a separate time shift operator on windows which means the WINDOW types need to be enhanced with time shifted windows in addition to the three basic ones introduced in this paper. As a result, all operations involving windows will need to be revisited to include time shifted windows. We plan to explore this alternative in our work in the near future.

Now we give several examples to illustrate these streaming operations. Let hurricane(name *string*, extent *sregion*) be a table of hurricanes with streaming spatial extent. Let county(name *string*, extent *region*) be a table of counties. Then the following statement continuously emits the names of the counties that intersect with hurricane K any time during the past 2 hours:

```
SELECT c.name
FROM   hurricane h, county  c
WHERE  duration(at(intersect(h.extent[past 2 hours], c.extent), TRUE)) <> 0
       AND h.name = K
```

Please note that the **intersect** operation returns a streaming bool type for the past two hours and is continuously updated. The **at** operation returns the time intervals where the value of intersect is TRUE. Then the duration (it is a shorthand for **rangevalues** of time) returns the duration of the time when the intersect operation is true. If this duration is not 0, then the county name is returned as it intersects with hurricane K some time during the past 2 hours.

Now let us look at the lifting of another spatial predicate **intersection** which returns the intersection part of two regions. The following statement returns the county name together with the intersection of the extent of the county and the current extent of the hurricane K continuously for all counties that intersect with hurricane K:

```
SELECT c.name, intersection(h.extent[now], c.extent)
FROM   hurricane h, county  c
WHERE  intersect(h.extent[now], c.extent) AND h.name = K
```

For a streaming data type and a temporal data type, the spatial predicate is applied to a pair of spatial extents from the two data types respectively with a time shift. For example, let hurricaneA(name *string*, extent *mregion*) be the archived hurricane information for the past years where *mregion* represents a moving region data type. The following statement is used to retrieve the names of the hurricanes whose current spatial extent has at least 80% overlap with a hurricane at the same time a year ago:

```
SELECT h.name
FROM   hurricane h, hurricaneA ha
WHERE  area(intersection(h.extent[now], ha.extent, 1 year))
       >80% * area(h.extent[now])
```

The operation between two streaming data types with a time shift can be illustrated by the following example. Let heavyRain(name *string*, extent *sregion*) be a table representing the heavy rain regions monitored in real time. Let flashFlood(name *string*, extent *sregion*) be a table representing flash flood. Then we can use the following statement to find the flash floods with heavy rain proceeding them two hours before:

```
SELECT ff.name
FROM   flashFlood ff, heavyRain hr
WHERE  duration(at(intersect(ff.extent[past 2 hours], hr.extent[past 4 hours], 2 hours)) <> 0
```

In the above statement, the **intersect** statement is applied between the extent of a flash flood and the extent of a heavy rain two hours before.

5 Language Embedding of the Stream Data Type

We have already seen several examples of language embedding of the STREAM data type and operations into SQL. To summarize, the stream data type is typically associated with an object. The object has static properties, e.g. name, and is modeled as a tuple in a table. In the FROM clause of a SQL statement, a table with streaming attribute can participate. The combination of stream operations among streaming attributes of the tables that result in a boolean TRUE or FALSE can appear in the WHERE clause. Objects that satisfy all the predicates including the streaming predicates in the WHERE clause will be returned as the results. All streaming operations and their combinations can appear in the SELECT clause and the results are subject to the condition that the associated objects satisfy the predicates in the WHERE clause. The following statement summarizes this paragraph.

```
SELECT STREAM operation, ...
FROM   table with streaming attribute, ...
WHERE  STREAM predicate, ...
```

To answer the three queries introduced in the introduction of the paper, we use the following static and dynamic spatial objects:

forestFire(firename: *string*, evaArea: *sregion*)
flood(floodname: *string*, extent: *sregion*)
house(owner: *string*, location: *point*)
road(roadname: *string*, extent: *line*)

The first query (**Q1**) regarding houses currently in the mandatory evacuation area of a forest fire may be answered with the following statement:

```
SELECT h.address
FROM   house h, forestFire ff
WHERE  distance(h.location,ff.evaArea [now])<50;
```

The second query (**Q2**) regarding houses traversed by a flood in the past 2 days may be answered with the following statement:

```
SELECT h.address
FROM   house h, flood f
WHERE  inside(h.location,traversed(f.extent [past 2 days]))
```

The third query (**Q3**) regarding roads immersed in flood for more than the past 24 hours may be answered by:

```
SELECT r.name
FROM   road r, flood f
WHERE  duration(deftime(at(inside(r.extent,f.extent [past 1 day]),TRUE)))=1 day
```

6 Conclusion and Future Work

In this paper, we proposed new data types and operations to support geo-streaming applications. We discussed their semantics and their embedding into a database language such as SQL. The proposed system is compatible with a data type based spatio-temporal database system. The system can be used to support continuous real-time queries on evolving spatio-temporal extents through the combination of these new data types and operations with SQL. Many real time monitoring applications will benefit from such a system.

This work is just the first step towards a full-fledged geo-stream database management system. We plan to first investigate the discrete representations for each new stream data type. For example, efficient data models and structures are needed to represent a streaming point in the past two hours.

Work also needs to be done in choosing efficient data structures and algorithms for query optimization. Please note that such queries may involve multiple geo-streams or a combination of geo-streams and spatio-temporal data types. For example, in streaming applications, indexing schemes are less sufficient since the cost of building the indexes and maintaining them will be high. Kinetic data structures deal with the fundamental geometric problems such as the maintenance of a convex hull from dynamic data. Algorithms were proposed to maintain geometrics and kinetic properties such as the closest pairs (to avoid collision) and minimal spanning tree (to maintain connections among moving sensors). In many cases, memory-based-kinetic data structures [10] will need to be considered for geo-stream applications instead.

Acknowledgment. We thank the anonymous reviwers of GIScience for their valuable insights, suggestions, and very detailed comments and corrections on this paper.

References

1. Tinydb, http://telegraph.cs.berkeley.edu/tinydb/
2. Abadi, D.J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: a new model and architecture for data stream management. The VLDB Journal 12(2), 120–139 (2003)
3. Ali, M.H., Aref, W.G., Bose, R., Elmagarmid, A.K., Helal, A., Kamel, I., Mokbel, M.F.: Nile-pdt: a phenomenon detection and tracking framework for data stream management systems. In: VLDB 2005: Proceedings of the 31st international conference on Very large data bases. VLDB Endowment, pp. 1295–1298 (2005)
4. Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M.J., Hellerstein, J.M., Hong, W., Krishnamurthy, S., Madden, S.R., Reiss, F., Shah, M.A.: Telegraphcq: continuous dataflow processing. In: SIGMOD 2003: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, p. 668. ACM Press, New York (2003)
5. Chen, J., DeWitt, D.J., Tian, F., Wang, Y.: Niagaracq: a scalable continuous query system for internet databases. SIGMOD Rec. 29(2) (2000)
6. Considine, J., Li, F., Kollios, G., Byers, J.: Approximate aggregation techniques for sensor databases. In: Proceedings of the 20th International Conference on Data Engineering (2004)
7. Deshpande, A., Guestrin, C., Madden, S.R., Hellerstein, J.M., Hong, W.: Model-driven data acquisition in sensor networks. In: Proceedings of VLDB, pp. 588–599 (2004)

8. Forlizzi, L., Güting, R.H., Nardelli, E., Schneider, M.: A data model and data structures for moving objects databases. In: SIGMOD 2000: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp. 319–330. ACM Press, New York (2000)
9. Grumbach, S., Rigaux, P., Segoufin, L.: The dedale system for complex spatial queries. In: SIGMOD 1998: Proceedings of the 1998 ACM SIGMOD international conference on Management of data, pp. 213–224. ACM Press, New York (1998)
10. Guibas, L.: Kinetic data structures: A state of the art report. In: The 3rd Workshop on Algorithmic Foundations of Robotics (1998)
11. Güting, R.H., Böhlen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N.A., Schneider, M., Vazirgiannis, M.: A foundation for representing and querying moving objects. ACM Trans. Database Syst. 25(1), 1–42 (2000)
12. Güting, R.H., de Almeida, V.T., Ansorge, D., Behr, T., Ding, Z., Höse, T., Hoffmann, F., Spiekermann, M., Telle, U.: Secondo: An extensible dbms platform for research prototyping and teaching. In: ICDE 2005: Proceedings of the 21st International Conference on Data Engineering, pp. 1115–1116. IEEE Computer Society, Los Alamitos (2005)
13. Guting, R.H., Schneider, M.: Moving Objects Databases . Morgan Kaufmann, San Francisco (2005)
14. Jain, N., Amini, L., Andrade, H., King, R., Park, Y., Selo, P., Venkatramani, C.: Design, implementation, and evaluation of the linear road bnchmark on the stream processing core. In: SIGMOD 2006: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pp. 431–442. ACM Press, New York (2006)
15. Mokbel, M.F., Aref, W.G.: Sole: scalable on-line execution of continuous queries on spatio-temporal data streams. VLDB Journal (accepted for publication, 2008)
16. Mokbel, M.F., Xiong, X., Aref, W.G.: Sina: scalable incremental processing of continuous queries in spatio-temporal databases. In: SIGMOD 2004: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pp. 623–634 (2004)
17. Relly, L., Röhm, U.: Plug and play: Interoperability in concert. In: Včkovski, A., Brassel, K.E., Schek, H.-J. (eds.) INTEROP 1999. LNCS, vol. 1580, pp. 277–291. Springer, Heidelberg (1999)
18. Systems, S.: StreamBase Server, http://www.streambase.com/
19. Tucker, P.A., Maier, D., Sheard, T., Fegaras, L.: Exploiting punctuation semantics in continuous data streams. IEEE Transactions on Knowledge and Data Engineering 15(3), 555–568 (2003)
20. Yiu, M.L., Mamoulis, N., Bakiras, S.: Retrieval of spatial join pattern instances from sensor networks. In: SSDBM, p. 25 (2007)

Turn to the Left or to the West: Verbal Navigational Directions in Relative and Absolute Frames of Reference

Toru Ishikawa[1] and Mika Kiyomoto[2]

[1] Graduate School of Interdisciplinary Information Studies & Center for Spatial Information Science, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
ishikawa@csis.u-tokyo.ac.jp
[2] Department of Urban Engineering, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

Abstract. This study examined how people use verbal route directions given in relative and absolute frames of reference in real-world navigation, particularly differences or similarities in cognitive load posed by the two frames of reference. Participants, Japanese speakers, walked the first set of five routes with relative (or absolute) directions and the second set of five routes with absolute (or relative) directions. For the first set of routes, participants performed equivalently on navigating the routes with relative and absolute directions, showing that they can somehow adapt to either way of thinking about space. But for the second set of routes, participants did better with relative directions than with absolute directions, showing that switching from a nonpreferred to a preferred frame of reference was easier than switching the other way around. In particular, participants with a poor sense of direction found the latter to be difficult. This asymmetric pattern of performance, depending on which frame of reference was used first, shows that the cognitive loads of processing information given in the two frames of reference are not the same. Concerning configurational understanding of the routes, participants did equally well with relative and absolute verbal directions, showing that they constructed equivalent mental images. These results were replicated in the second experiment of text comprehension, in which participants only read verbal directions and did spatial tasks. Implications of these results for the design of navigational aids were discussed.

Keywords: Spatial language, Frames of reference, Verbal navigational directions; Wayfinding, Cognitive maps.

1 Introduction

It is a daily, common experience for people to ask or to be asked questions about places; for example, "Where should we meet tomorrow?" "Where is that place?" and "How can I go there?" Responses to these questions can take various forms. One can show a printed map or draw a sketch map. Also, one can answer verbally, by providing descriptions such as "it is next to a tall brick building" or "the hotel is across from the post office" or even "you cannot miss it."

Thus information about space can be represented in different formats. Then, what is the characteristic of each presentation format and what effects does it have on the user's understanding and behavior? There is a body of literature about the understanding and use of maps and other spatial or visual representations, including 3-D visualizations, animations, and virtual environments (see Montello et al., 2004, for a review). Importantly, it has been shown that a significant portion of people, not only children but also adults, have difficulty using maps in the real world (Liben et al., 2002). Considering this fact, verbal descriptions, used on their own or supplementary to maps, could facilitate wayfinding and navigation. In fact, Streeter et al. (1985) found that their carefully constructed verbal directions were more effective than route maps for guiding drivers in an unfamiliar environment.

With the recent advance in information technologies, many kinds of navigational aids, notably devices equipped with global positioning systems (GPS), have been developed (e.g., Hightower & Borriello, 2001; Loomis et al., 2001; Shoval & Isaacson, 2006). On these navigation devices, speech guidance has been shown to be effective, as well as maps and other visual images (e.g., Goodman et al., 2004; Reagan & Baldwin, 2006). And there is much literature on the structure and contents of effective verbal route directions (e.g., Bradley & Dunlop, 2005; Denis et al., 1999, 2006; Hine et al., 2000; Klippel & Winter, 2005; Raubal & Winter, 2002).

Spatial language has attracted attention of researchers from many fields, including linguistics, psychology, and computer science. One of the most extensively discussed topics in spatial language is how the locations of objects in space are described and conveyed, namely the issue of *frames of reference*. Researchers have discussed different kinds of frames of reference, for example, egocentric and allocentric frames of reference (Klatzky, 1998), viewer-centered and object-centered frames of reference (Marr, 1982), and gaze, route, and survey frames of reference (Taylor & Tversky, 1996). On the basis of these discussions, Levinson (1996) proposed three kinds of frames of reference: intrinsic, relative, and absolute. An *intrinsic* frame of reference uses a coordinate system that is centered on an object, and the coordinates are determined by "inherent features" of the object, such as its shape or characteristic motion (e.g., the front of a car). A *relative* frame of reference uses a coordinate system fixed on a viewer, and describes the relations of figure and ground objects (e.g., left-right-front-back). An *absolute* frame of reference uses a coordinate system with a fixed origin on the ground (e.g., cardinal directions).

Research has shown that a specific language favors a specific kind of frame of reference in spatial description and spatial thinking. Levinson (2003) writes that "the frames of reference appropriately used in a language to describe specific situations are likely to correlate with the use of the same frames of reference in the nonlinguistic coding of the same scenes for memory and reasoning" (p. 171). For example, speakers of English, Dutch, or Japanese tend to use a relative frame of reference in spatial description and spatial thinking, and speakers of Tzeltal (a Mayan language spoken in highland Chiapas, Mexico) tend to use an absolute frame of reference in spatial description and spatial thinking. Thus, there seems to be a preferred frame of reference that speakers of a specific language find easy to use and a nonpreferred frame of reference that they find difficult.

On the other hand, there is also a research finding that people can construct equivalent mental images from both route and survey descriptions of environments, with all spatial information on the images being equally accessible (Taylor & Tversky, 1992). This seems on the surface to be inconsistent with the above findings from cross-linguistic studies, inasmuch as it suggests that a nonpreferred frame of reference does not yield poorer performance than a preferred one does.

To interpret these two lines of research findings, one needs to consider whether (a) spontaneous use of a specific frame of reference in linguistic or nonlinguistic coding and (b) comprehension of verbal directions given in different frames of reference are the same. One possibility is that people speaking a specific language use either of the two frames of reference in linguistic and nonlinguistic coding, but processing information given in the two frames of reference poses an equivalent level of cognitive load (i.e., spontaneous use and cognitive processing load are not related). Another possibility is that people use either of the two frames of reference in linguistic and nonlinguistic coding, and (or because) it poses a lower level of cognitive load than the other (i.e., which frame of reference is spontaneously used is correlated with the ease of processing information given in that frame of reference).

This research aims to investigate these possibilities in real-world navigation and text comprehension, with participants who speak Japanese, a language in which although all three frames of reference are familiar, a relative or intrinsic frame of reference is used predominantly in linguistic and nonlinguistic coding (Levinson, 2003). In both real-world and text-reading settings, verbal route directions given to half the participants were switched from *relative* to *absolute*, and those given to the other half of participants were switched from *absolute* to *relative*.

If the two frames of reference pose an equivalent level of cognitive load, participants would do equally well with either frame of reference. If a nonpreferred, absolute frame of reference poses a larger cognitive load, participants would do worse with it than with a preferred, relative frame of reference both before and after a switch of frames of reference (i.e., symmetric pattern of performance). Or it is also possible that participants' performance after a switch of frames of reference would be different depending on which frame of reference was used first (i.e., asymmetric pattern of performance). These possibilities were not specifically addressed in Taylor and Tversky's (1992) study, because the order of presenting relative and absolute descriptions was randomized across participants.

In the first experiment of real-world navigation, we looked at participants' wayfinding performance and configurational understanding of routes. In the second experiment of text comprehension, we looked at participants' response time and configurational understanding of described routes. In both experiments, we also examined whether participants' performance was correlated with their individual-difference measure of sense of direction.

2 Experiment 1: Outdoor Navigation

2.1 Method

2.1.1 Participants

Thirty-two college students (7 men and 25 women) participated in the experiment. Their ages ranged from 18 to 24, with a mean of 19.8 years. These participants were unfamiliar with the university campus used as the study area.

2.1.2 Materials

(a) Study area and routes

We used as the study area a campus of the University of Tokyo (Fig. 1). On the campus, we chose 10 places and the routes that connected these places. The routes were 61-524 m long, and along each route we selected distinct landmarks to be used in verbal route directions (explained below). On each route, the goal was not visible from the starting point, or vice versa.

(b) Verbal route directions

We created two kinds of verbal directions for each route. One kind of route directions used a relative frame of reference, with egocentric terms such as left or right. The other

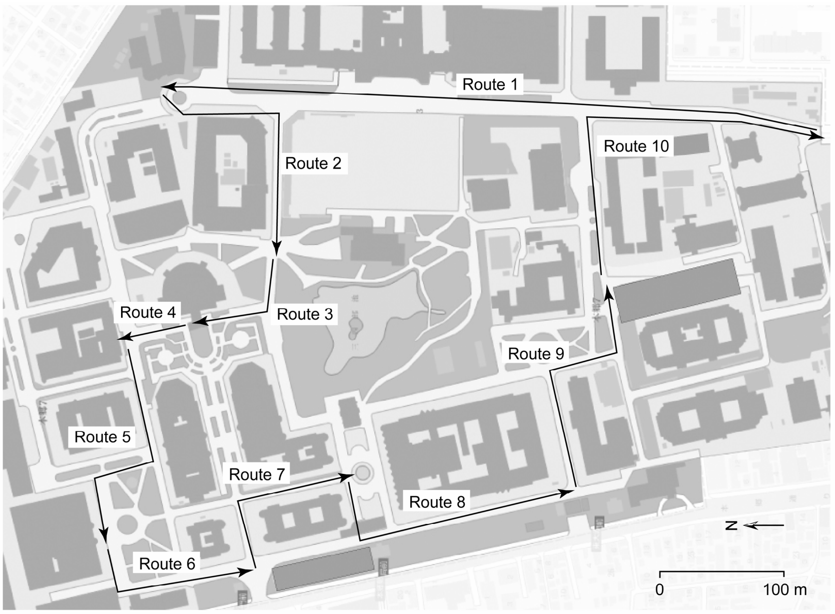


Fig. 1. Map of the study area. Ten routes were selected in the area (Routes 1-10). Arrows indicate the direction of travel.

kind of route directions was given in an absolute frame of reference, using cardinal directions (NSEW). In relative route directions, physical objects (buildings or intersections) were used to give distance; in absolute route directions, distance was given in meters.

As an example, relative route directions for Route 5 (Fig. 1) were "Go straight to the left. You see a square [open space] to the left. Go uphill and when you pass one building, turn right and go straight. You see a statue in a square to the left. When you pass the square, turn left and go straight. You see an entrance to the Engineering I Building to the right."

Absolute route directions for this route were "Go straight to the west for 110 m. You see a square to the south. When you hit a road running north-south, turn north and go straight for 60 m. You see a statue in a square to the west. When you hit a road running east-west, turn west and go straight for 50 m. You see an entrance to the Engineering I Building to the north."

Participants walked each route by reading verbal directions written on a postcard-sized card (10×14.8 cm), described in either a relative or an absolute frame of reference. The number of landmarks selected as "subgoals" on each route was the same for the two kinds of verbal directions (i.e., the two verbal directions shown above for Route 5 consists of six sentences).

(c) Self-report sense of direction

Participants filled out the Santa Barbara Sense-of-Direction Scale, which consists of fifteen 7-point Likert-type questions (Hegarty et al., 2002). Seven of the questions are stated positively (e.g., "I am very good at giving directions") and the other eight negatively (e.g., "I very easily get lost in a new city"). Hegarty et al. showed that people identified as having a good sense of direction on this scale did well on tasks that required configurational understanding of the environment ("survey tasks"), as opposed to tasks that required sequential understanding of landmarks and routes ("route tasks"). We included this individual-difference measure as a possible correlate with participants' performance on route navigation, especially comprehending their locations in an absolute frame of reference.

(d) Direction estimation task

At the goal of each route, participants estimated the direction to the starting point. To do that, participants were given a sheet of paper on which a circle with a radius of 5.5 cm was drawn, and drew a line from the center of the circle to indicate the direction to the starting point.

2.1.3 Design

Participants were randomly assigned to one of two groups concerning the order in which the two kinds of verbal route directions were given. Half the participants walked the first set of five routes with relative route directions and the second set of five routes with absolute route directions (called the *relative-absolute group*). The other half walked the first set of five routes with absolute route directions and the second set of five routes with relative route directions (called the *absolute-relative group*).

2.1.4 Procedure

At the beginning of the experiment, participants filled out the Sense-of-Direction Scale. Then they were individually taken to the starting point of the first route. At the starting point of each route, participants were shown a card on which verbal directions for the route were written, in either a relative or an absolute frame of reference, and instructed to go to the goal by following the verbal directions. The experimenter walked behind the participants without making conversation, and recorded the time that they took to walk the route, where they deviated from the route, and where they made stops. If they did not reach the goal within a time limit, which was determined as twice as long as the time that the experimenter took to walk the route, the experimenter took them to the goal by following the correct route. At the starting point of the route that participants walked for the first time with absolute route directions (Route 6 for the relative-absolute group, and Route 1 for the absolute-relative group), the experimenter showed the direction of north to participants by pointing in that direction. Only one participant asked, only once, for the direction of north again during the walk.

At the goal of each route, participants estimated, by facing straight ahead, the direction to the starting point, and assessed the difficulty of following the route on a 5-point scale. Participants finished all these experimental tasks within 90 min.

2.2 Results

2.2.1 Sense of Direction

For each participant, we calculated the mean of their answers to the 15 sense-of-direction questions. We reversed their answers to positively stated questions so that a higher score means a better sense of direction. There was no significant difference in sense of direction between the two groups of participants (with an alpha level of .05; same in the analyses below).

2.2.2 Travel Time, Distance, and Speed

We examined the time that participants took, and the distance that they traveled, to go from the starting point to the goal, for the first set of five routes and for the second set of five routes. A longer travel time or distance indicates that participants made stops during the walk or went off the routes described by the verbal route directions.

For the first set of five routes, there was no significant difference in either travel time or travel distance between the two groups. For the second set of five routes, participants took a longer time and traveled a longer distance with absolute route directions (197 s and 1,176 m) than with relative route directions (175 s and 1,082 m), Fs(1, 30) = 10.42 and 10.25, respectively, p's < .01.

Next, we examined participants' mean walking speed across the first set of five routes and across the second set of five routes, to see how smoothly participants walked the routes described by the verbal route directions. In a mixed ANOVA with the route set (first vs. second set of five routes) as a within-subject factor and the group (relative-absolute vs. absolute-relative group) as a between-subject factor, no significant main effects or interactions were found.

2.2.3 Number of Stops

We recorded when participants stopped during the walk for 15 s or longer, as an indication that they were trying to get oriented in space by comprehending the verbal route directions and relating them to the surrounding space. We calculated for each participant the total number of stops made on the first and second sets of five routes, and analyzed them in a mixed ANOVA with the route set as a within-subject factor and the group as a between-subject factor.

The main effect of route set was significant, $F(1, 30) = 40.32, p < .001$, showing that participants made more stops on the first set of five routes than on the second set of five routes. When we examined participants' wayfinding performance on the 10 routes, we found that they did poorly on Route 2. On that route, 24 (75%) participants made at least one stop and 25 (78%) participants went off the route at least once. These numbers were more than twice as large as those for the other routes (on this route, finding the path around the rotary at the starting point was difficult).

When we conducted a mixed ANOVA with Route 2 excluded, we found a significant main effect of route set, plus a significant interaction of route set and group, $F(1, 30) = 23.76, p < .001$, and $F(1, 30) = 11.28, p < .01$, respectively. (Between-group comparisons yielded the same results with and without Route 2.) Post hoc comparisons showed that there was no significant difference between the two groups for the first set of five routes, but for the second set of five routes participants in the relative-absolute group (i.e., with absolute route directions) made more stops than those in the absolute-relative group (i.e., with relative route directions), $F(1, 30) = 10.43, p < .01$ (Fig. 2A).

2.2.4 Deviations from the Routes

We next examined two measures concerning participants' deviations from the verbally described routes: (a) how often participants went off the routes and (b) how far they traveled off the routes, for the first and second sets of five routes.

In a mixed ANOVA, the main effect of route set was significant for both measures, Fs(1, 30) = 61.60 and 53.64, respectively, p's < .001. As in the analysis of the number of stops, a mixed ANOVA without Route 2 yielded a significant main effect of route set plus a significant interaction of route set and group (Fs[1, 30] = 18.86 and 10.14, respectively, p's < .001 and .01 for the number of route deviations; Fs[1, 30] = 9.42 and 8.27, respectively, p's < .01 for distance traveled off the routes). Post hoc comparisons showed that there was no significant difference between the two groups for the first set of five routes, but for the second set of five routes participants in the relative-absolute group (i.e., with absolute route directions) went off the routes more frequently and walked a longer distance off the routes than those in the absolute-relative group (i.e., with relative route directions), Fs(1, 30) = 7.94 and 11.33, respectively, p's < .01 (Figs. 2B, 2C).

2.2.5 Direction Estimates

For each participant, we calculated the mean absolute error of direction estimates across the first five routes and across the second five routes. We examined the effects of route set and group on direction estimates in a mixed ANOVA, and found no significant main effects or interactions. This study was not designed specifically to

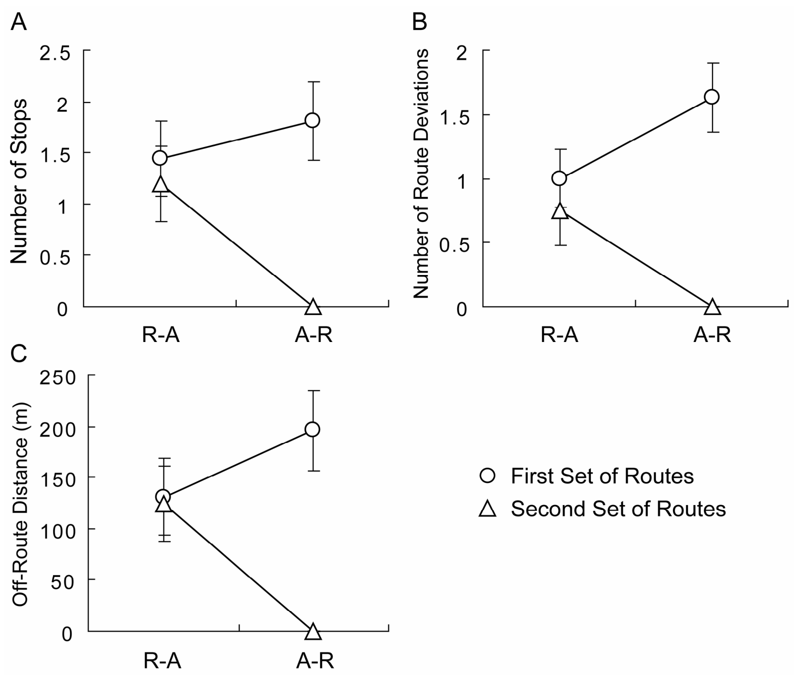


Fig. 2. Participants' performance on walking the verbally described routes: (A) the number of stops made on the routes, (B) the number of times that participants went off the routes, (C) the distance that participants traveled off the routes. Vertical lines depict standard errors of the means. For all measures, participants did worse with absolute verbal directions than with relative directions for the second set of routes. R-A = relative-absolute group; A-R = absolute-relative group.

examine sex-related differences, but there was a significant male-female difference only for this measure, with men making smaller direction errors than women.

2.2.6 Relationships with Self-Report Sense of Direction

Three of our wayfinding-performance measures (the number of stops, the number of route deviations, and distance traveled off the routes) were significantly correlated with participants' self-report sense of direction, when they walked the second set of routes with absolute verbal directions. Participants with a better sense of direction tended to make fewer stops ($r = -.57$, $p < .05$), went off the routes less frequently ($r = -.55$, $p < .05$), and walked a shorter distance off the routes ($r = -.59$, $p < .05$).

3 Experiment 2: Text Comprehension

To examine the difficulty of comprehending the verbal navigational directions and constructing mental images of the described routes, we next conducted an experiment in which participants only read verbal route directions and did spatial tasks.

3.1 Method

3.1.1 Participants

A new group of 22 people (15 men and 7 women) participated in the experiment. Their ages ranged from 21 to 58, with a mean of 26.7 years. These participants did not know about the first experiment of outdoor navigation.

3.1.2 Materials

We selected three routes each from the first and second sets of five routes used in the first experiment (Routes 2, 3, 5, 6, 7, and 9), by eliminating simple routes consisting mainly of straight walks, and created relative and absolute verbal route directions for each route. These verbal directions were the same as those used in the first experiment, except that the specific names of buildings were replaced by more common terms (e.g., a bank, bookstore, or station), so that participants would not realize that the verbal texts described the university campus (if they were familiar with it).

As an example, relative verbal directions for the route described in section 2.1.2(b) were "Go straight to the left. You see a square to the left. Go uphill and when you pass one building, turn right and go straight. You see a *bank* to the left. When you pass the bank, turn left and go straight. You see a *bookstore* to the right."

Absolute verbal directions for that route were "Go straight to the west for 110 m. You see a square to the south. When you hit a road running north-south, turn north and go straight for 60 m. You see a bank to the west. When you hit a road running east-west, turn west and go straight for 50 m. You see a bookstore to the north."

3.1.3 Design

Participants were randomly assigned to one of two groups concerning the order in which the two kinds of verbal route directions were given. Half the participants read relative verbal directions for the first set of three routes and absolute verbal directions for the second set of three routes (called the *relative-absolute group*). The other half read absolute verbal directions for the first set of three routes and relative verbal directions for the second set of three routes (called the *absolute-relative group*).

3.1.4 Procedure

At the beginning of the experiment, participants filled out the Sense-of-Direction Scale. Then, participants read verbal directions for each route, printed separately on an A4-sized sheet of paper, in either a relative or an absolute frame of reference. After reading the verbal route directions, they estimated the direction from the goal to the starting point and assessed the difficulty of the task on a 5-point scale, as participants in the first experiment did. Participants finished all these experimental tasks within 30 min.

3.2 Results

3.2.1 Sense of Direction

We calculated for each participant the mean of their answers to the 15 sense-of-direction questions. There was no significant difference in sense of direction between the two groups of participants.

3.2.2 Time to Complete the Tasks

We examined the time that participants took to read the verbal directions and do the experimental tasks, for the first set of three routes and for the second set of three routes. As in the first experiment, we conducted a mixed ANOVA with the route set (first vs. second set of three routes) as a within-subject factor and the group (relative-absolute vs. absolute-relative group) as a between-subject factor.

There was a significant main effect of route set, showing that participants took a longer time for the first set of three routes than for the second set of three routes, $F(1, 20) = 18.13, p < .001$. But at the same time, the interaction of route set and group was also significant, $F(1, 20) = 11.08, p < .01$, indicating that for the second set of three routes participants in the relative-absolute group (i.e., with absolute verbal directions) took a longer time than those in the absolute-relative group (i.e., with relative verbal directions) (Fig. 3).

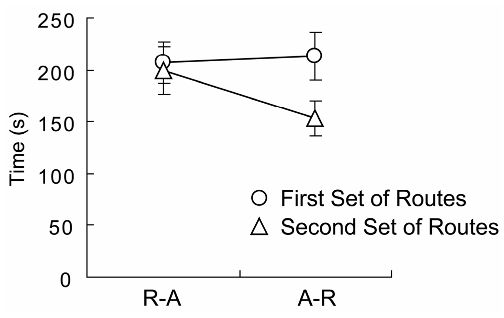


Fig. 3. Time to read the verbal directions and to complete the tasks. Vertical lines depict standard errors of the means. For the second set of routes, participants took a longer time with absolute verbal directions than with relative directions.

3.2.3 Direction Estimates

For each participant, we calculated the mean absolute error of direction estimates across the first set of three routes and across the second set of three routes. We examined the effects of route set and group on direction estimates in a mixed ANOVA, and found no significant main effects or interactions.

3.2.4 Relationships with Self-Report Sense of Direction

In the relative-absolute group, participants' self-report sense of direction was significantly correlated with the time to complete the tasks and ratings of task difficulty. For both the first and the second sets of three routes, participants with a better sense of direction took a longer time (r's = .72 and .73, p's < .05). For the second set of three routes (i.e., with absolute verbal directions), participants with a better sense of direction rated the difficulty of the tasks as easier than those with a poorer sense of direction ($r = .77, p < .01$).

4 Discussion

This study looked at how people use verbal route directions given in two kinds of frames of reference, relative and absolute, in real-world navigation. In particular, we examined the differences or similarities in the difficulty of using and processing information given in preferred and nonpreferred frames of reference, and whether people were able to adapt to or switch between the two frames of reference. Participants in this study were Japanese speakers, so their preferred frame of reference is *relative*. However, we found for the first set of routes that our participants performed equivalently on navigating the routes with relative and absolute verbal directions. This shows that when people are given these two kinds of verbal route directions and required to use them in navigation, they can somehow adapt to either way of thinking about space.

But this equivalence in performance disappeared for the second set of routes: Participants did worse with absolute route directions than with relative route directions. Thus, participants performed differently depending on which frame of reference was used, or activated in working memory, for the first set of routes. This shows that people find it easier to switch from a nonpreferred, absolute frame of reference to a preferred, relative frame of reference, than to switch the other way around. In particular, participants with a poor sense of direction find the latter to be difficult. At the same time, participants performed with absolute route directions as well as they did for the first set of routes with relative and absolute route directions. Thus, people can adapt to nonpreferred, absolute way of thinking at least at the same level of performance as when using a preferred, relative frame of reference.

The asymmetric pattern of change in performance, particularly facilitation of performance with a switch of frames of reference from a nonpreferred to a preferred one, shows that the cognitive loads of processing the two kinds of descriptions are not the same. The cognitive loads posed by the two frames of reference are somehow correlated with preferences inherent in a language for frames of reference in linguistic and nonlinguistic coding (as indicated by the better performance with relative route directions than with absolute route directions for the second set of routes). But it is not the case that a nonpreferred frame of reference always poses a larger cognitive load (performance was equivalent with relative and absolute route directions for the first set of routes).

On the other hand, concerning the configurational understanding of the routes, the two frames of reference did not cause a difference, for either the first or the second set of routes. As Taylor and Tversky (1992) discussed, people seem to be able to construct equivalent mental images from relative and absolute descriptions. That is, the difference lies not in mental images created but in the time needed or the ease with which to process information given in these two different reference frames, especially when frames of reference are changed from a preferred to a nonpreferred one. In absolute route directions, distance was given in meters, compared to the number of buildings or intersections in relative route directions, but it did not cause a difference in configurational understanding.

And importantly, these findings were replicated in the second experiment of text comprehension, in terms of response time. So the observed asymmetric pattern of performance is due to the difficulty with processing and comprehending verbal route directions given in the two different frames of reference. This difficulty should lead to more errors in navigating through the environment by linking verbally described landmarks and physical landmarks, especially for people with poor sense of direction.

A difference was observed between the two experiments in configurational understanding of the routes. The overall mean direction error for the first experiment (31.6°) was smaller than that for the second experiment (55.5°). This is due to the difference in the source of information in the two experiments. Participants in the first experiment walked and directly experienced the routes in the environment, while participants in the second experiment only read verbal directions and imagined walking the routes, thus lacking proprioceptive information (e.g., Couclelis & Gale, 1986; Klatzky et al., 1998).

These findings have a number of implications for the development of location-based systems and the design of speech guidance for navigational aids. First, mixing relative and absolute descriptions in verbal navigational directions does not seem to be a good idea, as it requires the user to switch between the two frames of reference back and forth, particularly from a preferred to a nonpreferred frame of reference (e.g., one guide book has the description "if you go out of the university campus and go north on A Street and east on B Street, you will see a stone monument on the left"). But if seen from a different perspective, exposure to mixed frames of reference could potentially be used as training of environmental learning ability or sense of direction. Miller and Allen (2001) suggested that switching frames of reference facilitated response time in judging the movement of a model car. A major difference between their study and our study is whether people thought about an object's movement as an outside viewer or people reasoned about their own movement.

In terms of spatial reasoning, it should be pointed out that in both relative and absolute route directions, information about direction was given categorically, and participants adapted to it at least to an acceptable degree. This kind of qualitative reasoning is an important characteristic of human spatial cognition (e.g., Frank, 1996). People do not normally require precise metric information in their everyday behavior, which is reasonable in light of the limited capacity of human memory.

We note that verbal descriptions used in this study were fixed at the scale of a university campus. It should be interesting to see how people respond to verbal route directions at different spatial scales (see Kataoka, 2005; Taylor & Tversky, 1996). Also, the characteristics of the environment, particularly the structure of the represented space such as grid versus radial patterns of streets or alignment of streets with cardinal directions, should affect the ease with which to use an absolute frame of reference (e.g., Montello, 1991; Richter & Klippel, 2005). And finally, we hope to see in further investigation whether these findings are observed cross-linguistically.

Acknowledgments

We acknowledge comments from A. Okabe, Y. Asami, Y. Sadahiro, and three anonymous reviewers. We also thank O. Imai and E. Imai for their help with recruiting study participants.

References

Bradley, N.A., Dunlop, M.D.: An experimental investigation into wayfinding directions for visually impaired people. Personal and Ubiquitous Computing 9, 395–403 (2005)

Couclelis, H., Gale, N.: Space and spaces. Geografiska Annaler 68B, 1–12 (1986)

Denis, M., Michon, P.-E., Tom, A.: Assisting pedestrian wayfinding in urban settings: Why references to landmarks are crucial in direction-giving. In: Allen, G.L. (ed.) Applied spatial cognition, pp. 25–51. Erlbaum, Mahwah (2006)

Denis, M., Pazzaglia, F., Cornoldi, C., Bertolo, L.: Spatial discourse and navigation: An analysis of route directions in the city of Venice. Applied Cognitive Psychology 13, 145–174 (1999)

Frank, A.U.: Qualitative spatial reasoning: Cardinal directions as an example. International Journal of Geographical Information Science 10, 269–290 (1996)

Goodman, J., Gray, P., Khammampad, K., Brewster, S.: Using landmarks to support older people in navigation. In: Brewster, S., Dunlop, M. (eds.) Mobile human-computer interaction, pp. 38–48. Springer, Berlin (2004)

Hegarty, M., Richardson, A.E., Montello, D.R., Lovelace, K., Subbiah, I.: Development of a self-report measure of environmental spatial ability. Intelligence 30, 425–447 (2002)

Hightower, J., Borriello, G.: Location systems for ubiquitous computing. Computer 34(8), 57–66 (2001)

Hine, J., Swan, D., Scott, J., Binnie, D., Sharp, J.: Using technology to overcome the tyranny of space: Information provision and wayfinding. Urban Studies 37, 1757–1770 (2000)

Kataoka, K.: Variability of spatial frames of reference in wayfinding discourse on commercial signboards. Language in Society 34, 593–632 (2005)

Klatzky, R.L.: Allocentric and egocentric spatial representations: Definitions, distinctions, and interconnections. In: Freksa, C., Habel, C., Wender, K.F. (eds.) Spatial cognition, pp. 1–17. Springer, Berlin (1998)

Klatzky, R.L., Loomis, J.M., Beall, A.C., Chance, S.S., Golledge, R.G.: Spatial updating of self-position and orientation during real, imagined, and virtual locomotion. Psychological Science 9, 293–298 (1998)

Klippel, A., Winter, S.: Structural salience of landmarks for route directions. In: Cohn, A.G., Mark, D.M. (eds.) Spatial information theory, pp. 347–362. Springer, Berlin (2005)

Levinson, S.C.: Frames of reference and Molyneux's question: Cross-linguistic evidence. In: Bloom, P., Peterson, M., Nadel, L., Garrett, M. (eds.) Language and space, pp. 109–169. MIT Press, Cambridge (1996)

Levinson, S.C.: Space in language and cognition. Cambridge University Press, Cambridge (2003)

Liben, L.S., Kastens, K.A., Stevenson, L.M.: Real-world knowledge through real-world maps: A developmental guide for navigating the educational terrain. Developmental Review 22, 267–322 (2002)

Loomis, J.M., Golledge, R.G., Klatzky, R.L.: GPS-based navigation systems for the visually impaired. In: Barfield, W., Caudell, T. (eds.) Fundamentals of wearable computers and augmented reality, pp. 429–446. Erlbaum, Mahwah (2001)

Marr, D.: Vision. Freeman, New York (1982)

Miller, C.R., Allen, G.L.: Spatial frames of reference used in identifying direction of movement: An unexpected turn. In: Montello, D.R. (ed.) Spatial information theory, pp. 206–216. Springer, Berlin (2001)

Montello, D.R.: Spatial orientation and the angularity of urban routes: A field study. Environment and Behavior 23, 47–69 (1991)

Montello, D.R., Waller, D., Hegarty, M., Richardson, A.E.: Spatial memory of real environments, virtual environments, and maps. In: Allen, G.L. (ed.) Human spatial memory, pp. 251–285. Erlbaum, Mahwah (2004)

Raubal, M., Winter, S.: Enriching wayfinding instructions with local landmarks. In: Egenhofer, M.J., Mark, D.M. (eds.) Geographic information science, pp. 243–259. Springer, Berlin (2002)

Reagan, I., Baldwin, C.L.: Facilitating route memory with auditory route guidance systems. Journal of Environmental Psychology 26, 146–155 (2006)

Richter, K.-F., Klippel, A.: A model for context-specific route directions. In: Freksa, C., Knauff, M., Krieg-Brückner, B., Nebel, B., Barkowsky, T. (eds.) Spatial cognition IV, pp. 58–78. Springer, Berlin (2005)

Shoval, N., Isaacson, M.: Application of tracking technologies to the study of pedestrian spatial behavior. Professional Geographer 58, 172–183 (2006)

Streeter, L.A., Vitello, D., Wonsiewicz, S.A.: How to tell people where to go: Comparing navigational aids. International Journal of Man-Machine Studies 22, 549–562 (1985)

Taylor, H.A., Tversky, B.: Spatial mental models derived from survey and route descriptions. Journal of Memory and Language 31, 261–292 (1992)

Taylor, H.A., Tversky, B.: Perspective in spatial descriptions. Journal of Memory and Language 35, 371–391 (1996)

Effect of Neighborhood on In-Network Processing in Sensor Networks

Muhammad Jafar Sadeq and Matt Duckham

Department of Geomatics, The University of Melbourne, Victoria, 3010, Australia
m.sadeq@pgrad.unimelb.edu.au, matt@duckham.org

Abstract. Wireless sensor networks are growing from a few hand-placed devices to more large-scale networks in terms of coverage and node density. For various concerns, such as scalability, larger network sizes require some management of the large volume of data that a sensor network delivers. One way to manage this data is processing information in the network. This paper investigates how a sensor network's network architecture (specifically, the neighborhood structure) can influence the conclusions that a sensor network makes from its measurements. The results demonstrate that non-planar structures are infeasible for routing and some in-network processing applications. Structures with low average edge lengths give better quantitative results, while those with high edge densities give better qualitative results.

1 Introduction

Wireless sensor networks (WSNs) are untethered networks of miniaturized sensor-enabled computers. WSNs are increasingly being used for geospatial applications, such as environmental monitoring. WSNs today typically use a periodic "sense-and-transmit" approach to transmit temporal snapshots of the entire network's readings to a centralized computing system [1][2]. The centralized computer system (such as a conventional GIS) is then tasked with analyzing the snapshots to determine what significant events have occurred. As such, many of today's WSN deployments can be regarded as sophisticated data loggers [3]. There are at least three important drawbacks to this approach:

- *Energy resources*: WSNs are highly resource constrained systems, in particular with respect to sensor node energy resources. Wireless communication is one of the most energy-intensive activities of a sensor node, so continually relaying data to a central system can dramatically shorten the useful lifetime of a WSN.
- *Information overload*: The fine-grained spatiotemporal detail becoming available from larger sensor networks means that individual data items become less and less meaningful. Transmitting all data can lead to high levels of redundancy and ultimately information overload.
- *Sensor/actuator networks*: The results of the analysis of sensor network data are often required by the network itself in order modify the behavior of

the network (e.g., activate or deactivate sensors to adapt the granularity of monitoring depending on the activity of the environment, sometimes called *georesponsive* sensor networks [4]). Removing information from the network, processing it centrally, then returning it to the network is an inefficient drain on network resources.

A growing proportion of current research in WSNs is dedicated to addressing these problems using *in-network processing*, where sensor nodes themselves collaborate in a decentralized manner to perform partial or complete processing of sensor data. This paper investigates how the structure of the network (links between nodes) can influence the results of in-network processing of spatial information and events.

Due to issues such as scalability, in-network processing often involves distributed processing with nodes using information from their *neighbors*. The question then arises as to which nodes are considered neighbors of a particular node. Often, the simplest solution is taken, where a node has as its neighbors all nodes that it can communicate with directly [5][6]. However, for routing, neighborhood will sometimes need to be limited to planar graphs (as discussed further in Sect. 2). Therefore, multiple definitions of neighborhood may exist in the same WSN. This paper evaluates the use of different neighborhood structures for in-network processing. Many of these structures are already used in routing research, thus gaining multiple uses out of a single infrastructure.

The importance of neighborhood structures to spatial applications can be seen when we realize that, for most environmental variables (such as temperature or light intensity), nodes take only point readings (with exceptions such as cameras). There is often no knowledge of the readings *between* nodes, either in the WSN or in any central server. Neighborhood structures may be used to estimate these unsampled regions. Techniques such as Voronoi diagrams [7] can partition the field into coverage areas in order to interpolate readings for the unsensed areas. Techniques such as kriging [8] estimate values at unsampled *points*. Given sufficient point estimates, point readings can be interpolated to a field. These techniques, however, may be computationally expensive when a node needs to do frequent interpolations in response to a dynamic environment.

From a qualitative spatial reasoning perspective, a fundamental interpolation process concerns the presence or absence of a boundary between neighboring nodes. Consider a binary region, for example a hot region (temperature$> 25°$C) and its complement (temperature$\leq 25°$C). If two nodes are connected topologically, and one senses that it is in the hot region and the other senses that it is out of the hot region, it is certain that there exists a boundary of the hot region boundary between them. If they both sense the hot region, it can be inferred that there is no boundary between them. An example is shown in Fig. 1. Based on a purely local, qualitative knowledge observations in their immediate neighborhood, Nodes A and D deduce that there must be a boundary between. Similarly, Nodes A and C can infer that there is no boundary between them. Thus, by answering a qualitative question ("Is there a boundary between myself and my neighbor?"), the network topology can be used to build a picture of a region.

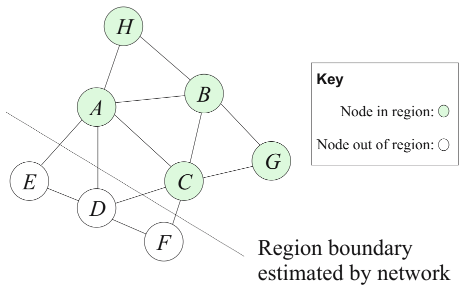


Fig. 1. Use of network edges to build a region

In order to evaluate various neighborhood structures, this paper quantitatively tests the ability of a WSN to approximate the boundary of a region when different structures are applied to it. Boundary was chosen as an evaluation criterion because it is a fundamental construct for the detection of spatial events in region evolution — events occur at a region's boundary or events create new boundaries. Boundary has salience for human spatial cognition and can be used to represent various spatial properties of the region such as its shape or its area. This work also tests the neighborhood structures' ability to qualitatively detect topological events occurring to a region.

It has been found that non-planar structures are infeasible not only for routing but can also cause inconsistencies in a WSN's knowledge and state. Neighborhood structures with a lower average edge length gives better quantitative results. Those with high edge densities give better qualitative results. Higher edge densities usually increase the average edge length, so high edge densities result in poorer quantitative performance.

2 Background

A lot of research effort has been given to developing neighborhood structures for WSNs because of their importance to routing. It is especially important to *greedy routing*, where a message is sent to a node geographically closer to the destination than the current node. An example of a greedy routing algorithm is Greedy Perimeter Stateless Routing (GPSR) [9].

Greedy forwarding can fail if the message reaches a "void" where no node is closer to the destination than the current node. GPSR therefore includes procedures to route around the edge of the void (Geographic Face Routing) [9][10]. Such forms of geographic routing (routing to a specific location) therefore require specific properties from the network graph of a WSN. One is that the number of voids must be minimized because face routing is more costly than greedy forwarding. Another is that the graph must be planar (there must be no crossing edges), otherwise face routing can fail. This in turn requires that there are no uni-directional links [10].

Researchers use network graphs to test their algorithms and protocols. Occasionally, researchers compare various neighborhood structures for the purposes of routing. However, to our knowledge, there is no research on the effect of the various neighborhood structures on the results of in-network processing. This may be because there is little research so far into in-network processing of higher level spatial information that rely on neighborhood structures.

This paper investigates how well different neighborhood structure approximate the boundary of a region and how well they detect topological events. A number of boundary detection algorithms have been proposed in the literature. In the classifier approach, nodes estimate the location of a straight line representing the local portion of the region boundary [5]. A node's distance from that line is used to determine whether it is a boundary node. In another approach, a node compares the average of the readings from its neighborhood to a threshold in order to determine whether it is on the boundary [6].

By contrast, this paper adopts a qualitative approach where boundary nodes are identified purely from their relationship to immediate neighbors (i.e., whether or not they neighbor a node on the opposite side of a boundary). In addition to simplifying the model and computation, adopting this qualitative approach can help improve the network's robustness to imperfection in sensor data and boundary indeterminary.

3 Neighborhood Structures

Modeling a network as a graph $G = (S, E)$, with a set of vertices $S \subset \Re^2$ representing sensor nodes as points in the plane and edges $E \subseteq S \times S$ representing the potential for direct one-hop communication between nodes, the following neighborhood structures are commonly used.

- *Localized Delaunay triangulation*: The Delaunay triangulation $DT(S)$ is the unique triangulation of S such that no point in S is inside the circumcircle of any triangle in $DT(S)$ [11]. A localized Delaunay triangulation (LDT) is a graph resulting from attempting to build a Delaunay triangulation (DT) using limited information (information local to a node). LDT does not have a complete convex hull since some edges that exceed the communication range of the nodes are not included in the graph. Depending on the algorithm, node positions and communication ranges, LDT may not even be a triangulation.
- *Gabriel graph*: E includes edges (x, y) and (y, x) between two points $x \in S$ and $y \in S$ if the circle with xy as the diameter is free of other points in S [12]. $GG(S)$ is a subgraph of $DT(S)$.
- *Relative neighborhood graph*: E includes edges (x, y) and (y, x) if the lune (the intersection of the two circles with centers x and y and with xy as radius) is free of other points in S. $RNG(S)$ is a subgraph of $GG(S)$ [12].
- *Localized greedy triangulation*: The greedy triangulation $GT(S)$ of a set of points S is built by first creating a list of all edges between all nodes in the network. The shortest edge is removed from the list and it is added to the triangulation if it does not cross an edge that has already been inserted into the triangulation. This continues until all the edges have been exhausted or the

number of edges in the triangulation is $3 \times N - 6$, where $N = |S|$. Thus the greedy triangulation (GT) attempts to minimize total edge length in a greedy manner [11]. The localized GT (LGT), built by nodes using local information, is not guaranteed to be a complete triangulation, similar to LDT.

– *Unit disk graph*: The unit disk graph (UDG) is a non-planar graph where there is an edge between two points as long as the points are at most a unit distance apart. In a WSN, this is a graph where each node has edges to all nodes that it can communicate with.

The graphs can be compared in Fig. 2, which shows graphs built in a distributed manner by a WSN simulation.

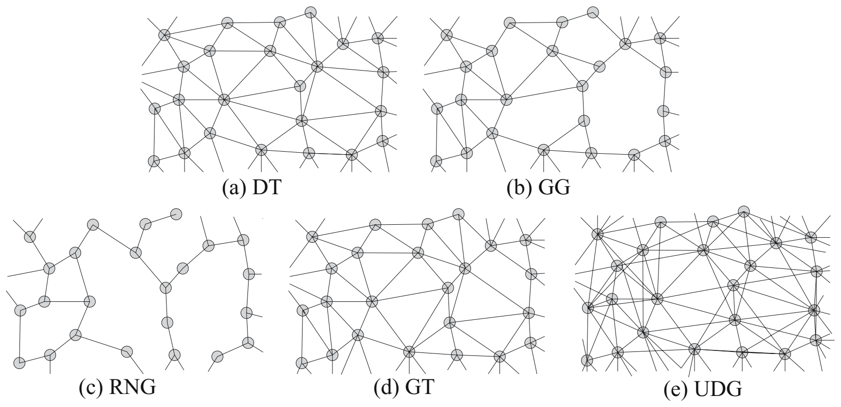


Fig. 2. Neighborhood structures built in a distributed manner

These five cover a variety of neighborhood structures. DT is well-known in spatial applications, so it is sensible to test a distributed version in a WSN. GG and RNG are built using completely local criteria and therefore lend themselves to implementation in WSNs using distributed processing. GT's usage of edge length as a criterion may make it suitable for WSNs because many localization algorithms determine distance (using, for example, received signal strength) *before* deriving coordinate information [13]. Finally, UDG is the most basic case where all possible edges between nodes exist.

4 Methodology

The neighborhood structures were compared against each other both quantitatively and qualitatively. The quantitative properties of the neighborhood structures were evaluated using a WSN's approximation of the boundary of a region. Their qualitative properties were tested by determining their ability to detect certain topological events. These tests were done by simulation.

4.1 Simulation Environment

It was infeasible to perform the tests using real sensor nodes due to the requirement for a large number of nodes to make the spatial properties of the neighborhood structures meaningful. Therefore, the tests were done by simulation. Repast (`http://repast.sourceforge.net/`), a Java-based agent modeling tool, was used. Each sensor node in the simulation was made an independent Java object. The field being simulated was also independent from the sensor nodes. The nodes interact among each other and with the environment in a decentralized manner.

4.2 Decentralized Computation of Neighborhood Structures

In our simulations, the neighborhood structures were built by the nodes themselves in a decentralized manner. Researchers have developed many decentralized algorithms to build network graphs that meet the routing requirements that have been mentioned in Sect. 2. The Cross-link Detection Protocol (CLDP) can generate a planar subgraph of any arbitrary graph of the network [10]. A localized DT built by a WSN can be ensured to be planar as long as a node uses information from nodes that are two communication hops away [14]. The Gabriel graph (GG) has also been used to develop network architectures [15], as it does not need a special algorithm to be built by sensor nodes locally.

In our simulations, the GG and RNG were built by simply by each node applying the GG and RNG rules (Sect. 3) to all nodes that it can communicate with directly. Each node $x \in S$ starts with a list of these nodes, which it can acquire in an initial handshaking phase. x tests each node in this list to check whether an edge between them is permitted in GG or RNG.

For the LDT, for every pair of nodes y and z in x's communication range, if there were no nodes in the circumcircle of $\triangle xyz$, edges (x, y) and (x, z) were added to E. Thus, for GG, RNG and LDT, the node does not compute the complete graph of all the points in its communication range.

If x uses nodes in its communication range to compute the LDT, a non-planar graph may result. This happens when x rejects a triangle $\triangle xyz$ due to a node w in $\triangle xyz$'s circumcircle, but y or z accepts $\triangle xyz$ because w is out of its communication range. This situation results in unidirectional links. If nodes that are two communication hops away are included in the triangulation procedure, planarization is ensured [14]). For simplicity, however, in our simulations x removed all unidirectional edges (each edge (x, y) for which y did not have an edge (y, x)). Additionally, triangles whose circumcircles include more than three nodes were not included in the graph because they can result in crossing links. These procedures gave us a planar graph, but not a complete triangulation. Rather, the LDT is a graph based on the rules for Delaunay triangulation.

LGT requires the nodes to compute all possible edges between all its neighbors and create the complete graph to ensure no edges cross. Although fast algorithms for GT exist in the literature, our objective in this work was not to use the most efficient algorithms but rather to test the neighborhood structures. Therefore, we used algorithms that were the simplest for us to implement, i.e., we applied the basic rules of the neighborhood structures.

Finally, UDG was built by each node creating an edge to each other node within.

4.3 Quantitative Test: L^2 Error Norm

To quantitatively determine the accuracy of the region boundaries as approximated by the WSN under various neighborhood structures, the shape, location and size of the approximated region boundary were compared to the actual boundary. The L^2 *error norm* provided a simple means to test all of these properties. The L^2 error norm is the area of the region enclosed between the boundaries of two shapes. A WSN monitoring a region O determines an approximation P of the original region's boundary (Sect. 5.1). The L^2 error norm was computed in this paper by finding the area of the symmetric difference between O and P as a proportion of the total area of O, i.e.,

$$L^2 \text{ error norm} = \frac{area((O-P) \cup (P-O))}{area(O)} \quad . \tag{1}$$

An L^2 error norm of zero means that not only are the areas of the two shapes equal, but also that their boundaries are in complete agreement.

4.4 Qualitative Test: Detecting Topological Events

For various reasons, including space limitations, of the six fundamental topological events for dynamic areal objects (Fig. 3) [16], only merge and split are investigated in this paper. In concurrent work, an algorithm has been developed to detect these topological events and distinguish them from each other and from non-topological events. It was found that the algorithm to detect *self-merge* is similar to that for split, *partial-split* required other events being detected correctly, and the effects of misidentified *appearances* and *disappearances* can be found in incorrect detection of other topological events. For these reasons, it was felt that merge and split were suitably representative of topological events to be used to analyze the neighborhood structures' effectiveness.

The complete algorithm to detect and distinguish spatial events is beyond the scope of these paper. In fact, the algorithm for detecting a merge is not needed to understand why a merge can be incorrectly detected, so that is left out too. Only a condensed version for the detection of split is presented.

Split Detection. The algorithm for detection of a split is complicated due to the difficulty in distinguishing it from contraction. In both Fig. 4(a) and Fig. 4(b), Node A has detected a change in the environment. Consequently, the region being monitored no longer includes A. To differentiate split and contraction, Node A maintains its neighbors in a list sorted by direction (called the *cyclic ordering* of the neighbors). With this sorted list, it determines how many blocks of region/hole surround it.

If there are two such blocks, the region has contracted — in Fig. 4(a), A is surrounded by an in-region block consisting of B, D and C and an out-of-region

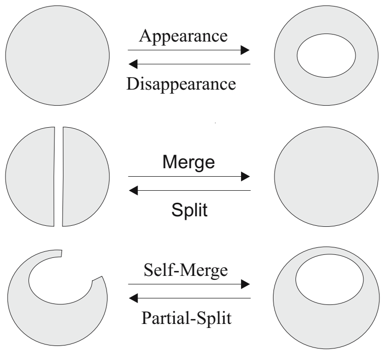


Fig. 3. Six fundamental topological events for dynamic areal objects

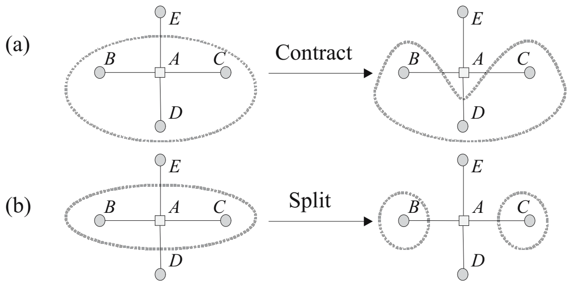


Fig. 4. Split/contract

block consisting of only E. If there are four or more blocks (Fig. 4(b): B, D, C and E are all in separate blocks around A), the region has split.

To demonstrate why four blocks are necessary, consider D. If D were not in A's neighborhood, A would not be able to determine whether a split has occurred. This is because, from A's point of view, there is a consecutive block of region around it consisting of B and C. It therefore assumes that a contraction has occurred. Additionally, the absence of D means that A cannot locally determine whether the region is connected beyond its immediate neighborhood.

Further cases (such as if D is actually in a hole in the region) are easily resolved, but those issues are beyond the scope of the paper. In our simulations to evaluate neighborhood structures, we controlled the region evolution so that only merge and split occurred. Therefore, in this paper, it is not necessary to discuss the complete algorithm. Instead, it is sufficient to realize that, to detect a split, a node must have at least four neighbors.

5 Boundary Approximation and Analysis

One hundred random deployments of 500 nodes were generated for the simulations to determine which of the neighborhood structures gave rise to the best boundary approximation from the WSN. For each of the hundred deployments, each of the neighborhood structures was applied.

5.1 Experiment 1: Polygonal Region

In a binary field, the WSN was used to monitor a simple polygonal region (Fig. 5). In order to determine the WSN's boundary approximation, a message is passed along the boundary of the region (usually the nodes just inside the boundary). The message collects a list of boundary coordinates as it moves from node to node along the region boundary. The message is passed anticlockwise along the region boundary — this was an arbitrary choice. The ordered list of coordinates that results from this message is the WSN's polygonal approximation of the actual region boundary.

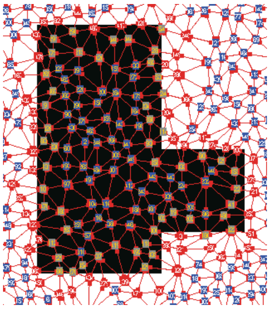


Fig. 5. Example of LGT-networked WSN monitoring a polygonal region

Figure 6 can be used to demonstrate how the WSN builds its polygonal approximation of the region boundary. The nodes maintain their neighbor tables sorted by cyclic ordering around the node. For example, B's anticlockwise neighbor list may be (I, A, C, H). Let B be the initiator of the boundary approximation task. From the ordered list of its neighbors, H is the first neighbor of B that is outside the region (since it directly follows a neighbor that is inside the region). B starts the boundary polygon with the midpoint of BH. It then adds midpoint of BI to the polygon. Since the next neighbor in the list, A, is inside the region, the list of coordinates is passed to A. A adds the midpoints of AJ, AK, AE and AD to the list, then passes it on to C. When the list of coordinates returns to B, it receives the complete estimate of the boundary polygon.

Results. Figure 7 (one of 100 simulations) gives a qualitative view of the WSN's approximation of the monitored region's boundary using the different neighborhood structures. Table 1 gives the average L^2 error norm for the 100 simulations for each neighborhood structure. The results show that RNG is the best at estimating a region boundary, while LGT is the worst.

T-tests were performed to determine whether the differences between the various neighborhood structures were statistically significant at the 5% level. The tests revealed that they were significantly different, even LDT and LGT, which were very close in results with a 0.001 difference in L^2 error norm.

Absence of UDG from Results. Observe that the results do not include the UDG. This is due to the nature of the message transmission scheme in Sect. 5.1

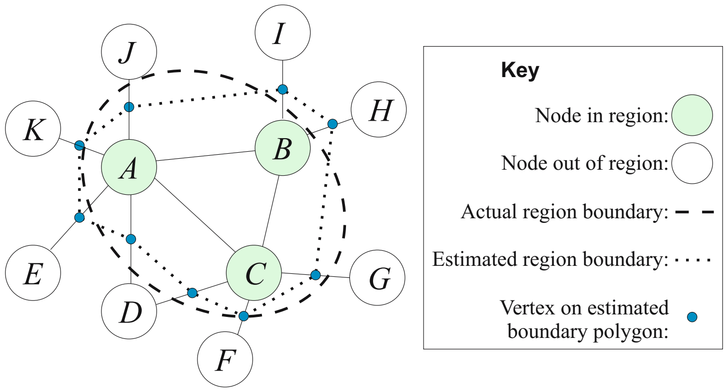


Fig. 6. A simple example of a WSN's polygonal estimate of a region boundary

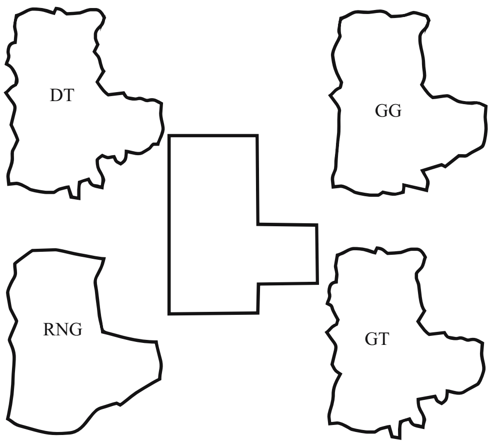


Fig. 7. Approximations of the polygonal region boundary using different neighborhood structures

(tracing around a boundary), which is a form of face routing since messages are sent along edges according to their cyclic order. This has the pitfalls of geographic face routing, and so non-planar graphs cause the message to loop forever. An example is shown in Fig. 8. A message arrives at D (1), is passed to B (2), then to A (3), C (4) and back to B (5), at which point it continues along path 3-4-5. Observe that removing either the edge DB or CA (making the graph planar) can solve the problem. The UDG is intrinsically non-planar, and so is unsuitable to the distributed boundary approximation algorithm above.

Discussion. It was expected that the neighborhood structures with higher edge density (such as LDT and LGT) would perform better because the higher number of edges would allow the network to trace around the boundary more accurately. However, the results showed that RNG performed the best.

Table 1. Average L^2 error norms for the polygonal region

Neighborhood Structure	Average L^2 error norm
LGT	0.106
LDT	0.105
GG	0.0971
RNG	0.0919

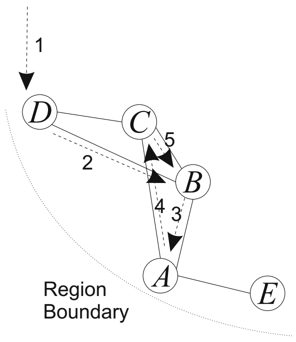


Fig. 8. Non-planar neighborhood causes an infinite loop

A reason for this unexpected result is that RNG is the sparsest of the four neighborhood structures. It therefore has fewer edges that intersect the region's boundary, and consequently the WSN's estimate of the region boundary has fewer vertices. This results in fewer and longer edges in the approximation polygon. This seemed a good explanation for why RNG was best at monitoring the simulated region of few vertices and long linear edges [5].

5.2 Experiment 2: Circular Region

The results of Experiment 1 suggested that shapes with linear edges were best monitored by a WSN with RNG. Therefore, experiment was repeated with a region without straight lines — the circular region in Fig. 9.

Results. Figure 10 is a sample of the shape of the boundary approximations due the the four neighborhood structures.

The L^2 error norm, averaged over 100 simulation runs, showed that RNG has the best results again (Table 2). The results were statistically significant at the 5% level, and they were also correlated — whenever one neighborhood structure work well, the others work well too (Pearson correlations of between 0.845 and 0.942). The exception was RNG, which was comparatively uncorrelated (Pearson correlations of between 0.578 and 0.674).

Discussion. Contrary to our expectations, RNG performed better in both circular and polygonal regions. An explanation for this is that RNG is sparser than

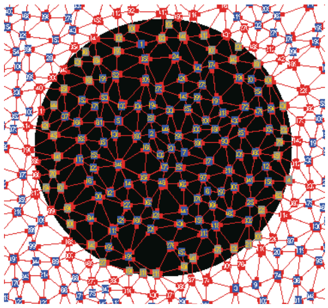


Fig. 9. Example of LGT-networked WSN monitoring a circular region

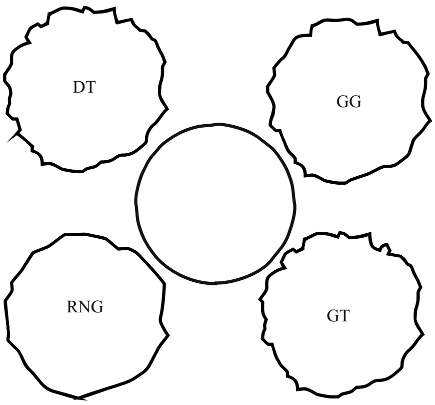


Fig. 10. Approximations of the circular region boundary using different neighborhood structures

the other structures in terms of edge density. Its edges are therefore shorter than those of GG and DT, of which it is a subgraph. Whenever one of the RNG edges intersect the region boundary, the likelihood is that any point on the edge is closer to the actual boundary location than any point on the longer edges of LDT. Due to this low average edge length of RNG, its boundary points were on average the most accurate.

Table 2. Average L^2 error norms for the circular region

Neighborhood Structure	Average L^2 error norm
LGT	0.0689
LDT	0.0683
GG	0.0618
RNG	0.0508

The results for the circular region were better than the polygonal region due to the sharp corners of the polygonal region. The WSN is not always able to estimate correctly these sharp turns of the polygon boundary.

6 Detection of Topological Events

It has been seen that the low edge density, and consequently the low average edge length, of RNG makes it suited to locating the boundary. The following experiments determine how the neighborhood structures perform in detecting topological events.

6.1 Experiment 3: Merge

In these experiments, a small square region component moves until it merges with the polygonal region component (Fig. 11). For each neighborhood structure, we ran the simulation with ten different random deployments of nodes. For each simulation run, we counted how many merges were detected.

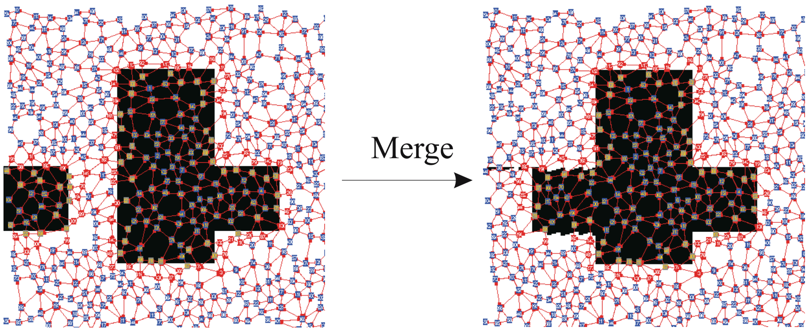


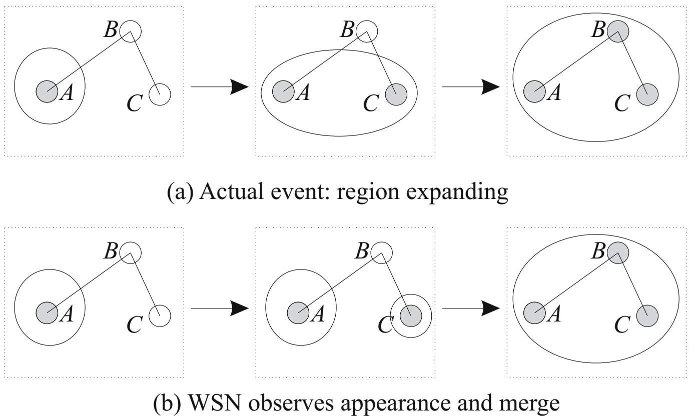
Fig. 11. Simulation of merge

Results. WSNs under LDT, GG and LGT detected exactly one merge corresponding to the merge event in Fig. 11. RNG, however, detected 12 additional merges (Table 3).

Table 3. Number of merges detected with each neighborhood structure in 10 simulation runs

Neighborhood Structure	Number of merges detected
LGT	10
LDT	10
GG	10
RNG	22

Discussion. Figure 12 demonstrates how a WSN may incorrectly identify a merge. Figure 12(a) shows a region expanding. Figure 12(b) shows the WSN's belief of what is occurring. Since A and C are disconnected, their detections of the region are treated as separate. So C believes that a new region component has appeared. When the region expands to include B, it sees two distinct region components, one at A and the other at C, come together, so it identifies it as a merge.



B
A
C
(a) Actual event: region expanding
(b) WSN observes appearance and merge

Fig. 12. False detection of merge

If the neighborhood structure had been denser, with an edge between A and C, C would have been able to correctly identify the change as an expansion due to having a neighbor that was in the region previously. Thus, the sparsity of the network graph causes false positives in detecting a merge. This is why RNG gives 12 false merge detections in addition to the 10 correct merges.

6.2 Experiment 4: Split

After the merge, the part of the region to the right splits from the region (Fig. 13). Ten different random placements of nodes were generated and then used to run simulations for each neighborhood structure. The number of splits detected by the WSN was recorded.

Results. Table 4 shows how many splits were missed by the WSN when the four neighborhood structures were applied. LDT and LGT did not miss any splits. GG missed 6 splits and RNG did not manage to detect a single split.

Discussion. Section 6.2 explains why a node needs to have at least four neighbors in order to detect a split. Figure 2 shows that with GG, nodes do not always have four neighbors. With RNG, there are even more nodes with the same problem. Therefore, although GG did sometimes successfully detect a split, RNG did not detect a single one. For a node at the pinch point where a split occurred, there were never four neighbors when RNG was used.

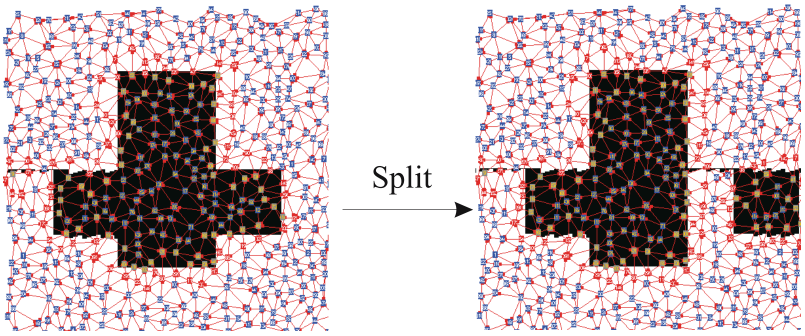


Fig. 13. Simulation scenario for detecting splits

Table 4. Number of missed splits with each neighborhood structure in 10 simulation runs

Neighborhood Structure	Number of splits missed
LGT	0
LDT	0
GG	6
RNG	10

A Further Reason for the Exclusion of UDG. Section 5.1 gives routing difficulties as the reason for the exclusion of UDG. A different reason for the inapplicability of UDG in Experiments 3 and 4 is that non-planar graphs result in inconsistencies in the WSN's knowledge if we use the qualitative approach in Sect. 1. In Fig. 14, if A and C have an edge between them, they assume that the region is continuous between them. If there is an edge between B and D, the region is discontinuous between A and C — it may be said that there is a hole in the region between B and D. If there are edges between all the nodes, from A's and B's points of view, the region is continuous, while to B and D, the region is discontinuous. This inconsistency in the WSN's beliefs causes problems when the nodes need to collaborate in various tasks. For these reasons, when we attempted to apply our algorithm to a UDG-networked graph, the network reported many false events and resulted in an incorrect network state. Therefore, we do not use UDG to detect topological events.

6.3 Summary

- It is not possible to use UDG in a WSN where routing and event detection is a concern, at least with the algorithms used in this paper.
- RNG gives the best quantitative results due to short average edge lengths, but it performs very poorly in detecting topological events. However, RNG performs poorly at qualitatively detecting topological events, to the point that it cannot be recommended for such tasks.

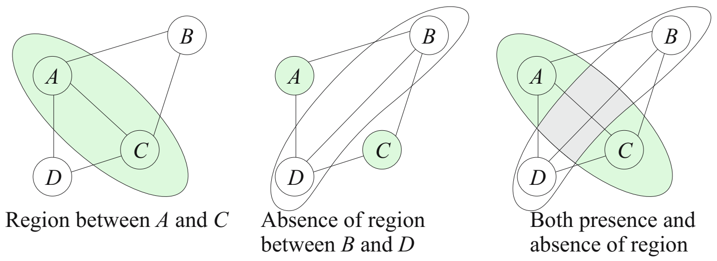

B
A
C
D
Region between A and C
B
A
C
D
Absence of region between B and D
B
A
C
D
Both presence and absence of region

Fig. 14. Inconsistency due to non-planar graph

– LDT and LGT give the best qualitative results for event detection due to having sufficient neighbors to distinguish complex topological and non-topological events.

7 Conclusion

Processing of data in a WSN to generate information is an important tool for information management for various reasons, such as to improve scalability and for in-network usage of higher level information. Neighborhood structures are important for the in-network processing of spatial information. The results demonstrate that the properties of the neighborhood structure can have a significant influence on the qualitative and quantitative observations of the WSN.

When network edges are used to give spatial context to sensor readings (such as build regions), the length of the edges affects the quantitative estimates made by the WSN. Longer edges give the WSN more opportunities to make mistakes about assumptions any estimated made along the edge. Even so, other than UDG, none of the neighborhood structures failed at quantitative tasks, so there is some flexibility in choosing an appropriate structure for a WSN when other factors are taken into account.

However, qualitative comparisons rely heavily on ordering of information. Therefore, the lack of sufficient information to be ordered can give rise to false results. For this reason, the RNG fails at the split detection task, while GG performs poorly. RNG also gives false positives in detecting merges.

LDT and LGT in particular perform alike in the qualitative tasks and LDT is slightly better at quantitative tasks. If there are no other considerations (such as computational complexity), LDT should always be chosen over LGT. As for the others, researchers and engineers need to choose a structure that best fits their applications.

One of the factors that will influence the choice of neighborhood structures is the cost of building and maintaining it as the node set changes due to, for example, sleep cycles, movement or failure. These costs will depend on the distributed algorithms that are employed. Since this paper does not investigate algorithms for distributed creation and maintenance of the various neighborhood structures, it is not possible as yet to provide a cost comparison of these structures.

In our current research, we are developing the algorithm (part of which was presented in Sect. 5) to detect qualitative spatial events. From the results of this paper, we will be using LDT for our future investigations.

Acknowledgments

Matt Duckham's research is supported by the Australian Research Council under ARC Discovery Grant DP0662906, and by a Joint Collaborative Research Grant funded by the University of Melbourne and the Victorian Department of Sustainability and Environment (DSE). Matt is also grateful for the support of the Ordnance Survey of Great Britain.

References

1. Mainwaring, A.M., Culler, D.E., Polastre, J., Szewczyk, R., Anderson, J.: Wireless Sensor Networks for Habitat Monitoring. In: Proc First ACM International Workshop on Wireless Sensor Networks and Applications, pp. 88–97 (2002)
2. Hamilton, M.P., Graham, E., Rundel, P.W., Allen, M.F., Kaiser, W., Hansen, M.H., Estrin, D.L.: New Approaches in Embedded Networked Sensing for Terrestrial Ecological Observatories. Environmental Engineering Science 24(2) (2007)
3. Hart, J.K., Martinez, K.: Environmental Sensor Networks: A Revolution in the Earth System Science? Earth-Science Reviews 78, 177–191 (2006)
4. Duckham, M., Nittel, S., Worboys, M.F.: Monitoring Dynamic Spatial Fields Using Responsive Geosensor Networks. In: GIS 2005: Proc. 13th Annual ACM International Workshop on Geographic Information Systems, pp. 51–60. ACM Press, New York (2005)
5. Chintalapudi, K., Govindan, R.: Localized Edge Detection in Sensor Fields. Ad Hoc Networks 1(2-3), 273–291 (2003)
6. Jin, G., Nittel, S.: NED: An Efficient Noise-Tolerant Event and Event Boundary Detection Algorithm in Wireless Sensor Networks. In: MDM 2006: Proceedings of the 7th International Conference on Mobile Data Management (MDM 2006), p. 153. IEEE Computer Society, Los Alamitos (2006)
7. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications. Springer, Heidelberg (1997)
8. Barnett, V.: Environmental Statistics: Methods and Applications. Wiley, Chichester (2004)
9. Karp, B., Kung, H.T.: GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In: MobiCom 2000: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, pp. 243–254. ACM Press, New York (2000)
10. Kim, Y.J., Govindan, R., Karp, B., Shenker, S.: On the Pitfalls of Geographic Face Routing. In: Proceedings of the Third ACM/SIGMOBILE International Workshop on Foundations of Mobile Computing, pp. 34–43. ACM Press, New York (2005)
11. Preparata, F.P., Shamos, M.I.: Computational Geometry: An Introduction. Springer, New York (1985)
12. Zhao, F., Guibas, L.: Wireless Sensor Networks: An Information Processing Approach. Elsevier/Morgan-Kaufmann, Amsterdam (2004)

13. Vaidyanathan Ramadurai, M.L.S.: Localization in Wireless Sensor Networks: A Probabilistic Approach. In: Zhuang, W., Yeh, C.H., Droegehorn, O., Toh, C.T., Arabnia, H.R. (eds.) Proceedings of the International Conference on Wireless Networks, pp. 275–281. CSREA Press, Las Vegas (2003)
14. Li, X.-Y., Calinescu, G., Wan, P.-J., Wang, Y.: Localized Delaunay Triangulation with Application in Ad Hoc Wireless Networks. IEEE Transactions on Parallel and Distributed Systems 14(10), 1035–1047 (2003)
15. Funke, S., Milosavljevic, N.: Infrastructure-Establishment from Scratch in Wireless Sensor Networks. In: Prasanna, V.K., Iyengar, S.S., Spirakis, P.G., Welsh, M. (eds.) DCOSS 2005. LNCS, vol. 3560, pp. 354–367. Springer, Heidelberg (2005)
16. Jiang, J., Worboys, M.F.: Event-Based Topology for Dynamic Planar Areal Object. International Journal of Geographical Information Science, Under review (2008)

Similarity-Based Information Retrieval and Its Role within Spatial Data Infrastructures

Krzysztof Janowicz[1], Marc Wilkes[1], and Michael Lutz[2]

[1] Institute for Geoinformatics, University of Muenster, Germany
janowicz, marc.wilkes@uni-muenster.de
[2] European Commission – Joint Research Centre
Institute for Environment and Sustainability, Ispra, Italy
michael.lutz@jrc.it

Abstract. While similarity has gained in importance in research about information retrieval on the (geospatial) semantic Web, information retrieval paradigms and their integration into existing spatial data infrastructures have not been examined in detail so far. In this paper, intensional and extensional paradigms for similarity-based information retrieval are introduced. The differences between these paradigms with respect to the query and results are pointed out. Web user interfaces implementing two of these paradigms are presented, and steps towards the integration of the SIM-DL similarity theory into a spatial data infrastructure are discussed. Remaining difficulties are highlighted and directions of further work are given.

1 Introduction and Motivation

Semantics-based information retrieval [1,2] plays an increasing role in GIScience and research on the geospatial semantic Web. In general, two approaches can be distinguished, those based on classical subsumption reasoning and those based on so-called non-standard inference techniques [3,4] – similarity being one of them. While the number of similarity theories for information retrieval is increasing, the number of real geo-application is still low. So far, the most prominent reason was that existing theories were not able to handle the expressivity of description logics used by ontologies on the (geospatial) semantic Web. Recently developed theories [5,6,7,8] bear the potential to close this gap, which moves the focus towards new challenges. This paper addresses one of these challenges: How can similarity be integrated within existing spatial data infrastructures (SDIs) to support users during information retrieval?

The paper is based on the SIM-DL similarity theory [7], which is introduced in section 2. This section also describes relevant background in the areas of similarity measurement, description logics, and SDIs. Sections 3 and 4 present the two main contributions of the paper. In section 3, we present a classification of semantics-based information retrieval paradigms. We investigate the differences between subsumption and similarity-based approaches in terms of how a query is phrased and which results can be expected. The examined paradigms are grouped into intensional and extensional approaches (and combinations of both). In section 4, we discuss how the similarity-based paradigms can be implemented using a SDI and what difficulties have to be overcome. Section 5 concludes the paper and points out directions for future research.

In order to illustrate our work, the following scenario is used: A tourist, Nicole, is planing a trip to Utah, US. As a passionate canoeist she would like to spend some time canoeing. Since she has never been to Utah before, she does not know any waterbodies that might please her. However, she does remember some rivers and lakes that she canoed a few years ago while visiting friends in Canada. Therefore, she decides to search for this kind of features browsing services on the Web. We assume that these services are part of a SDI that also includes a catalogue service providing information about available geographic feature types and a web similarity service (WSS) based on the SIM-DL server [7].

2 Related Work

This section introduces related work. The objectives and basic components of SDIs are presented in section 2.1. Sections 2.2 and 2.3 describe similarity measurement and description logics, respectively. These provide the basis for the SIM-DL similarity theory introduced in section 2.4.

2.1 Spatial Data Infrastructures

The main goal of SDIs is to offer access to distributed data sources. The development of interoperability specifications – and here most prominently the work within the Open Geospatial Consortium[1] (OGC) – has created a technology evolution that moves from standalone GIS applications towards a more loosely coupled and distributed model based on self-contained, specialized, and interoperable (Web) services [9]. In such an infrastructure, where resources are distributed and controlled by different organizations, catalogue services provide a means for describing the services' locations and capabilities. They store metadata and support users in discovering and using these resources. The OGC has specified a catalogue service for the Web (CS-W) and related metadata profiles.

While thus the SDI concept promises an efficient sharing and reuse of geographic data among heterogeneous user groups [9,10], most existing SDIs are still at an early stage in their development. Many just offer geoportals that integrate on-line map viewers and catalogue services for their data holdings [11,12].

In this work, we focus on geographic data provided through the Web feature service interface (WFS) and organized into geographic feature types. In SDIs, metadata about feature types can be stored in so-called *feature catalogues* [13]. Feature catalogues define the types of features, their operations, attributes, and associations. Their goal is to provide a better understanding of geographic data

[1] The OpenGIS standards and specifications are available from `http://www.opengeospatial.org/standards/`.

in order to enable users to judge whether the data fits their purpose. An implementation of a feature type catalogue as an extension package for the ebRIM Profile of the CS-W has been proposed [14].

2.2 Similarity Measurement

Similarity has its origin in cognitive science and was established as a theory to investigate how entities are grouped into categories, and why some categories (and their members) are comparable while others are not [15,16]. *Semantic* similarity measures the proximity of meanings as opposed to purely structural comparison. While entities can be expressed in terms of attributes, the representation of concepts[2] is more complex. In dependence of the (computational) characteristics of the representation language, concepts are specified as unstructured bags of features[3], regions in a multidimensional space, or set-restrictions specified using various kinds of description logics. As the computational concepts are models of concepts in human minds, similarity depends on what is said (in terms of representation) about these concepts. Context is the next big challenge for similarity research. In most cases, meaningful notions of similarity cannot be established without defining in respect to what similarity is measured [16,18,19,20].

Similarity-based information retrieval plays an increasing role in GIScience. Based on Tversky's feature model [17], Rodríguez and Egenhofer [21] developed the Matching Distance Similarity Measure which offers a basic context theory, feature weights, and asymmetry, while Raubal [22] proposed geometric similarity measures based on conceptual spaces. Several measures [5,6,7,8] were developed to close the gap between ontologies specified in description logics and similarity theories which had not been able to cope with the expressivity of these languages. Other similarity theories [23,24] have been established to determine the similarity between spatial scenes and also investigate how to use similarity within spatial queries [24]. The ConceptVISTA [25] ontology management and visualization toolkit uses similarity for knowledge retrieval and organization.

2.3 Description Logics and Inference

Description logics (DL) are a family of knowledge representation languages used to model concepts and individuals within a knowledge base. A knowledge base consists of a TBox which contains the terminology, i.e., the concepts within a given domain, and an ABox which stores assertions (about named individuals). A DL system offers services to reason about the content of a knowledge base. Standard reasoning services include satisfiability, subsumption, and instance checking. The computation of the most specific concept (MSC) for an individual and the least common subsumer (LCS) of several concepts are so-called non-standard reasoning services [4]. Computing the MSC of an individual

[2] The term *concept* is used in this paper for the ontological representation of geographic feature types and should not be confused with the concepts in human minds.

[3] In the sense of concept characteristics, see [17].

yields the least concept that the individual is an instance of. Accordingly, the LCS of some concepts is the least concept that subsumes all of them, i.e., there is no subconcept of the LCS which is a superconcept of all these concepts.

Computing the MSC and LCS can serve a variety of purposes. A top-down approach for constructing knowledge bases is not always feasible, since not all relevant concepts might be known beforehand. Instead, one could only specify the building blocks in the TBox and then introduce typical examples as individuals in the ABox. By computing the MSC (and LSC) for these individuals, more complex concepts can be added to the TBox. A semantics-based retrieval approach based on computing the LCS has been proposed by Möller et al [3].

2.4 SIM-DL Similarity Theory and Implementation

While the previous sections focus on similarity in general, this section gives an insight into a similarity theory (and its implementation) which measures similarity between concepts specified in expressive description logics.

SIM-DL [6,7] is an asymmetric and context-aware similarity measurement theory used for information retrieval. It compares a search concept C_s with a set of target concepts $\{C_{t_1}, ..., C_{t_m}\}$ from an ontology (or several ontologies using a shared top-level ontology). The concepts themselves can be specified using various kinds of expressive description logics. The compared-to target concepts can be either selected by hand, or derived from the so-called context of discourse $\mathcal{C}_d$ [6,19,18], i.e., a subset of the ontology, also referred to as the domain of application [21]. It is defined as the set of concepts which are subsumed by the context concept C_c ($\mathcal{C}_d = \{C_t | C_t \sqsubseteq C_c\}$). Hence, each (named) concept $C \in \mathcal{C}_d$ is a target concept for which the similarity $sim(C_s, C_t)$ is computed. Besides cutting out the set of compared concepts, $\mathcal{C}_d$ also influences the resulting similarities (see [6,7] for details). With respect to the canoeing scenario this means that the similarity between the concepts $River$ and $Canal$ also depends on whether C_c is set to $Watercourse$ or the more general $Waterbody$ (see figure 1). Up to now, the user has to specify the context concept manually. Information retrieval paradigms overcoming this restriction are discussed in section 3. SIM-DL offers an extended context model, but we focus on $\mathcal{C}_d$ here (see [18]).

SIM-DL compares two DL concepts in canonical form [6,26] by measuring the degree of overlap between their definitions. A high level of overlap indicates a high similarity and vice versa. DL concepts are specified by applying language constructors, such as intersection or existential quantification, to primitive concepts and roles – hence forming complex concepts. Consequently, similarity is defined as a binary and real-valued function $C_s \times C_t \rightarrow \mathrm{R}[0,1]$ providing implementations for all language constructs offered by the used description logics. Finally, the overall similarity between concepts is the normalized (and weighted) sum of the single similarities calculated for all parts of the concept definitions. A similarity value of 1 indicates that the compared concepts cannot be differentiated, whereas 0 indicates that they are not similar at all. SIM-DL is an asymmetric measure, i.e., the similarity $sim(C_s, C_t)$ is not necessarily equal to $sim(C_t, C_s)$. Therefore, the comparison of two concepts does not only depend on their descriptors, but also on the direction in which both are compared. In case of concepts composed by disjunction, SIM-DL distinguishes between two similarity modes, the maximum similarity and the average similarity. In the first case, similarity depends on the most similar concept that is part of the disjunction. In the second case, similarity is defined as the average of all involved concepts. While this distinction is a consequence of the used representation language and the definition of similarity functions in SIM-DL [6,7], we will discuss its importance for information retrieval in section 3.

A single similarity value computed between two concepts hides most of the important information. It does not answer the question whether there are more or less similar target concepts in the examined ontology. It is not sufficient to know that possible similarity values range from 0 to 1 as long as their distribution is unclear [7,18]. Besides these interpretation problems, isolated comparison puts too much stress on the concrete similarity value. It is hard to argue that and why the result is (cognitively) plausible without other reference values [18]. Consequently, SIM-DL focuses on similarity rankings. The result of a similarity query is an ordered list with descending similarity values. SIM-DL, supports various result representations including font-size scaling or categorization.

The SIM-DL theory is implemented as semantic similarity service (called WSS here). The current (beta) release[4] 2.2 supports subsumption reasoning and similarity measurement up to $\mathcal{ALCHQ}$, as well as MCS and LCS computation (up to $\mathcal{ALE}$). More details on the SIM-DL implementation and a similarity plug-in to the Protégé ontology editor are given by Janowicz et al. [7]. The extensions to the description logics communication interface DIG, necessary to integrate the WSS within the Semantic Web are discussed by Wilkes and Janowicz [27].

3 Intensional and Extensional Retrieval

In this section, we will introduce paradigms for similarity-based information retrieval and group them into intensional and extensional approaches (and combinations of both). We will motivate the need for similarity by contrasting it to approaches purely based on subsumption reasoning. To illustrate the differences, we define two concepts, the *intended concept* C_i, which represents exactly the information the user is looking for, and the *search concept* C_s, which is the concept actually used in the search. We make this distinction, because we assume that in most cases the intended concept is not part of the queried ontology and has to be approximated by the search concept. The result of a query, i.e., the concepts or individuals returned to the user, is the better the more accurate C_s approximates C_i. In the ideal case, C_s would be equal to C_i. If ${C_s}^{\mathcal{I}} \subset {C_i}^{\mathcal{I}}$, all instances of C_s fulfill the user's requirements. In contrast, if ${C_s}^{\mathcal{I}} \supset {C_i}^{\mathcal{I}}$, some instances of C_s might not fulfill the user's requirements. Consequently, a user interface should support users in defining appropriate search concepts. We further introduce the notions of target concepts C_t and instances I_t, which will be used in our descriptions of similarity-based search paradigms.

[4] The release can be downloaded at `http://sim-dl.sourceforge.net/`.

In the following, we assume that the information the user is searching for is represented as individuals and concepts within an ontology, and that she uses these artifacts as the basis for her search. How this representation can be mapped to features and feature types provided in a SDI is presented in section 4.

To illustrate the different approaches, we will use an excerpt of a hydrology ontology (figure 1). For readability, the figure shows a simplified representation of the ontology only including its is-a relations; more expressive DL statements (including concept defintions based on roles such as $hasFlowVelocity$ or $hasOwnership$) are not depicted. Note that such expressive statements can also be used in the proposed paradigms and SIM-DL [7].

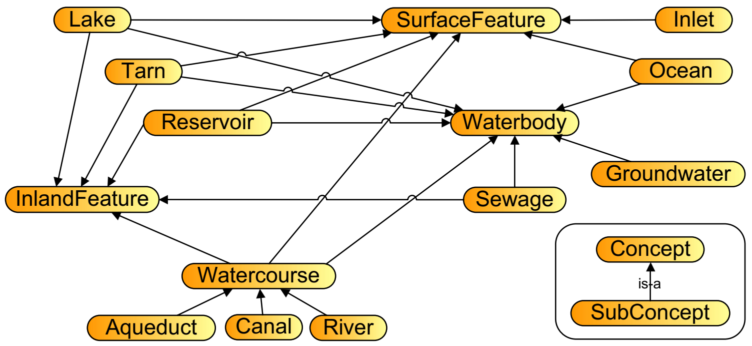


Fig. 1. Fragment of an hydrology ontology

Referring to the canoeing scenario, Nicole is searching for the ad-hoc category [28] of *canoe-able waterbodies* (C_i). As a corresponding concept is not available in the ontology, she needs to define a query using existing concepts. We assume a graded structure [28,29] for such ad-hoc categories, i.e., there are more typical and less typical canoe-able waterbodies (on concept and instance level).

3.1 Subsumption-Based Retrieval

In terms of subsumption-based retrieval and as depicted in figure 2, the following (intensional) approaches for deriving a query, i.e., the search concept, can be distinguished (see also [1]):

a) Search concept is a subconcept of C_i. The user can specify a particular search concept (e.g., $River$) that is known to be a subconcept of C_i. Consequently, while all individuals of C_s also satisfy the requirements for C_i, many appropriate individuals will not be captured as ${C_s}^{\mathcal{I}} \subset {C_i}^{\mathcal{I}}$.

b) Search concept is a superconcept of C_i. The user can specify a particular search concept (e.g., $Waterbody$) that is known to be a superconcept of C_i. Consequently, while all individuals of C_i will be returned, many inappropriate individuals (e.g., instances of $Tarn$, $Ocean$, $Sewage$, or $Groundwater$) will also be captured as ${C_s}^{\mathcal{I}} \supset {C_i}^{\mathcal{I}}$.

c) Search concept is defined by conjunction. The user can try to build a more appropriate search concept using a conjunction of existing named concepts (e.g., $SurfaceFeature \sqcap InlandFeature \sqcap Waterbody$). First, this would require a complex user interface. Second, without a detailed knowledge of the examined ontology, the relation between C_i and C_s remains unclear to the user. Additionally, there is a high likelihood to get an empty result set (because of the conjunction constructor). Thus, such an approach is more suitable for ontology engineers than for end-users.

d) Search concept is defined by disjunction. The user can select several concepts known to be subconcepts of C_i, hence forming C_s as disjunction of those concepts (e.g., $Canal \sqcup River \sqcup Lake \sqcup Reservoir \sqcup Inlet$). While this would require a complex user interface and is time consuming, it would result (if the concepts are carefully selected) in a good approximation of C_i.

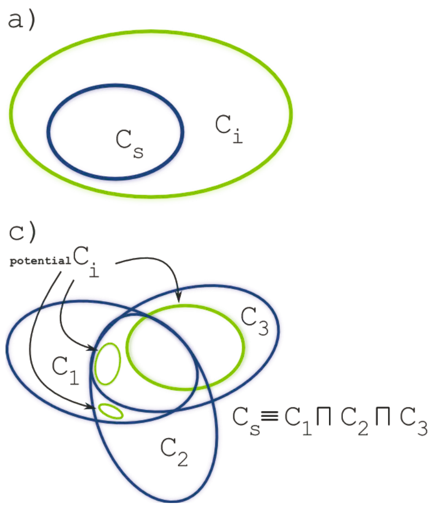


Fig. 2. Subsumption-based information retrieval paradigms

3.2 Similarity-Based Retrieval

In terms of similarity-based retrieval and as depicted in figure 3, the following approaches for deriving a query can be distinguished.

Intensional Paradigm. The intensional information retrieval paradigm exclusively relies on concept descriptions to reason about similarity, i.e., no individuals are taken into account. Consequently, a concept ranking is returned to the user.

e) Prototypical search concept. The user can specify a prototypical search concept C_s (e.g., $River$) such as described in approach a and define a context concept C_c (e.g., $Waterbody$) in addition, which is known to be a superconcept of C_i (as in b)[5]. All subconcepts of C_c, called target concepts C_t here,

[5] Approach e can also be modified to take concepts formed by disjunction or conjunction (such as in c and d, repsectively) into account.

are compared for similarity to C_s; see section 2.4. As we assume a graded structure, a decreasing similarity to the search concept is interpreted as less intended concept. Such a ranking can already be delivered back to the user [18,30]. While this approach is comparable to a combination of *a* and *b*, the ranking is a major advantage and supports the user in generating queries such as in case *d*, without requiring a detailed insight into the underlying conceptualizations. While the selection of a prototypical concept is less difficult, finding an appropriate context concept manually is more difficult. As each concept returned within the ranking is a subconcept of C_c, it follows that if C_c does not capture the minimum characteristics of C_i , inappropriate concepts may be returned (however, they would have a low position in the ranking due to their low similarity to the search concept). Subconcepts of C_c whose similarity is 0 (or below a pre-defined threshold) are *not* part of the ranking. Each concept C_t from the ranking satisfies the more probable the user's requirements, the higher its similarity is to C_s.

Additionally, one could automatically generate a new search concept C_{sn} as a disjunction of the returned similar concepts (above a certain threshold) which would be compareable to approach *d* without that the user needs to find and select those concepts by hand. Note that one cannot think of the instances as members of a fuzzy set with the similarity of their concepts (to C_i) as the degree of membership. In SIM-DL (and most related measures), inter-concept similarity cannot be directly mapped to inter-instance similarity (i.e., a similarity $sim(River, Canal)$ of 0.76 does not imply that the similarity between all rivers to all canals is 0.76).

Extensional Paradigm. The extensional information retrieval paradigm is a *query-by-example*, and hence relies exclusively on individuals to reason about similarity. A concept (the LCS) computed from the set of examples, called reference individuals here, is used to pre-select the compared-to target individuals. Consequently, the user's query is answered by returning ranked individuals.

f) Reference individuals (for individual similarity). The user can specify a set of reference individuals $\{I_{r_1}, ..., I_{r_n}\}$ (e.g., particular rivers and lakes). The retrieval of target individuals can then be subdivided into three steps. First, the most specific concept MSC_{r_i} for each reference individual is determined.The second step is the computation of the least common subsumer for $\{MSC_{r_1}, ..., MSC_{r_n}\}$. The LCS comprises those characteristics that are common to all MSCs, and is therefore used as the context concept C_c[6]. Finally, retrieving the instances of C_c yields the target individuals $\{I_{t_1}, ..., I_{t_m}\}$ (particular rivers, lakes, reservoirs, canals, etc.). Since all target

[6] Consider the following example: $MSC_{r_1} \equiv River \sqcap \exists hasFlowVelocity.Velocity \sqcap \exists hasOwnership.Public$ and $MSC_{r_2} \equiv Lake \sqcap \exists hasOwnership.Public$ results in the LCS $C_c \equiv SurfaceFeature \sqcap Waterbody \sqcap InlandFeature \sqcap \exists hasOwnership.Public$. The restriction $hasOwnership.Public$ is common to both MSCs, and although $Lake$ and $River$ do not match, their common superconcepts $SurfaceFeature$, $Waterbody$, and $InlandFeature$ are considered by the LCS.

individuals instantiate C_c, they share the same characteristics common to all reference individuals. Now, similarity is used to account for those characteristics that differ among the reference individuals. For example, a characteristic that is common to $\{I_{r_1}, ..., I_{r_{n-1}}\}$, but does not apply to I_{r_n}, is not captured by C_c, and is therefore no requirement for a target individual. Nevertheless, a target individual that shares that characteristic with $n-1$ reference individuals might be more relevant than target individuals lacking that property. The overall relevance of a target individual can be determined by comparing it to each of the reference individuals and combining (e.g., averaging) the resulting similarities. The result returned to the user is a ranking of target individuals illustrating their similarity to the reference individuals.

Recapitulating, the reference individuals are an extensional way to approximate C_i, and at the same time their common characteristics are used as context concept. The set of target individuals might capture inappropriate individuals as $\{I_{r_1}, ..., I_{r_n}\} \subset C_i{}^{\mathcal{I}} \subseteq \{I_t | I_t \in C_c \ and \ I_t \notin \{I_{r_1}, ..., I_{r_n}\}\}$ holds. Due to the similarity ranking of target individuals, this drawback is compensated. The higher a target individual is ranked the more likely it is within $C_i{}^{\mathcal{I}}$. In contrast to paradigm e, one could also think of the returned ranking as a fuzzy set by replacing the crisp membership (or instance-of) relation with its counterpart from fuzzy sets theory[7]. Then, the degree of membership of a certain (target) individual is given by its similarity value.

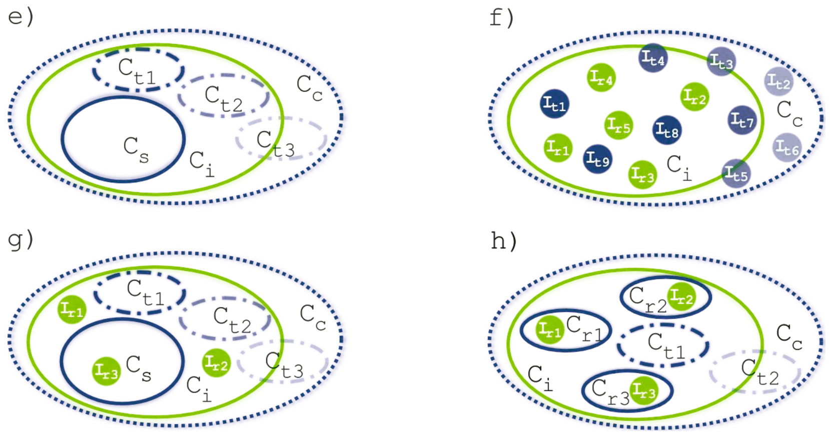


Fig. 3. Similarity-based information retrieval paradigms

Combinations of the Intensional and Extensional Paradigm. A combination of the intensional and extensional paradigms reduces the difficulties in selecting appropriate search and context concepts by allowing for the selection of reference individuals (however, both paradigms return (similar) concepts).

[7] See Cross and Sudkamp [31] for an overview on fuzzy sets and similarity.

g) Prototypical search concept *and* reference individuals. The user can specify a search concept (e.g., *River*) and a set of reference individuals (e.g., waterbodies Nicole had canoed before). This is a combination of the paradigms *e* and *f*, where the search concept is used for comparison and the least common subsumer of the reference individuals is used as context concept to define the context of discourse. The result is a concept ranking such as in *e*. The advantage is that the user does not need to specify the context concept manually. However, the distinction between search concept and reference individuals may be difficult to explain to the user.

h) Reference individuals (for concept similarity). The user can specify a set of reference individuals. Their LCS is computed and acts as context concept. All resulting target concepts ($C_{t_j} \sqsubseteq C_c$) are compared to the concepts (C_{r_i}) the reference individuals are instances of. For example, if one reference instance is a *Lake*, one a *River* and a third a *Canal*, the LCS would be $SurfaceFeature \sqcap Waterbody \sqcap InlandFeature$ and the target concepts all concepts from our ontology fragment except *Ocean*, *Inlet*, *Groundwater* and *Sewage*. Each target concept is compared to each C_r (*Lake*, *Canal*, and *River* in our example); note that each C_r is a subconcept of C_c, and therefore a target concept itself. This raises the question of how the resulting ranking should be generated. The similarity modes introduced for SIM-DL allow for two different solutions. Out of the user's reference individuals a search concept can be defined as disjunction of the respective reference concepts. Next, either the average or maximum similarity mode can be used to compute the similarity $sim(C_s, C_{t_j})$. In the first case, each C_t is compared to all C_r and the average of its similarity values is used for the ranking. In the second case, the ranking depends on the highest similarity value to one of the C_r. With respect to our example and the average mode, *River*, *Canal*, and *IrrigationCanal* would occupy a higher position in the ranking, followed by *Lake* (see figure 5). This is due to the two watercourses selected as reference individuals.

As in *g*, this approach is also a combination of *e* and *f*, but does not require the manual definition of the search concept. The user only needs to specify reference individuals while C_s and C_c are computed automatically.

3.3 Summary

In contrast to the subsumption-based approaches, similarity supports the user in phrasing queries and delivers a ranking to help the user in judging how well returned individuals or concepts fit her requirements. In the cases *a* and *d*, it is guaranteed that all returned concepts are subconcepts of the intended concepts, i.e., fulfill the user's requirements – this is not the case for similarity-based retrieval in general. To overcome this difficulty the context concept is introduced to capture the minimal characteristics and only its subconcepts are compared for similarity. The approaches *e*–*h* offer different solutions on how to reduce the effort of phrasing the search and the context concept, and automate these steps.

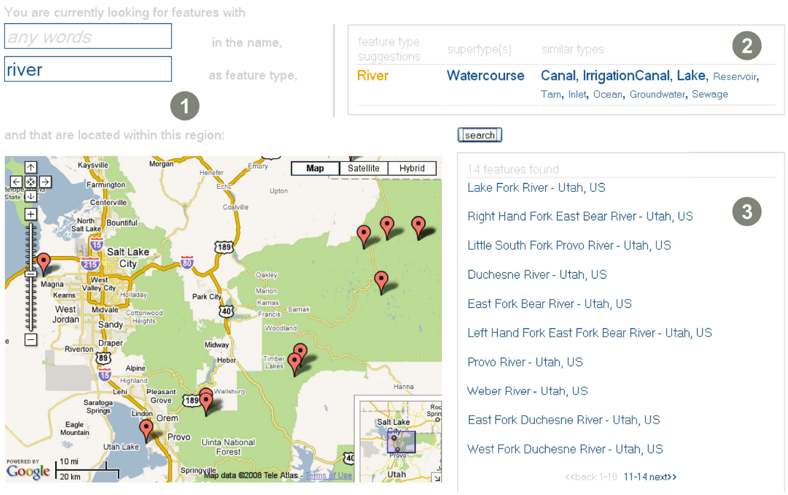
The question which of the proposed paradigms fits best depends on the application area. In general, the focus of similarity is more to facilitate the navigation and browsing through results and hence to improve interaction with the user. Section 4 introduces two prototypical user Web interfaces to demonstrate how similarity-based information retrieval can be integrated into a SDI.

4 Integration into SDI

This section presents two conceptual designs for Web user interfaces implementing the similarity-based retrieval paradigms *e* and *h* for the canoeing scenario[8]. An architecture and workflow for the integration of SIM-DL into an SDI is discussed and the requirements for such integration are pointed out.

4.1 Similarity-Enabled User Interfaces

Figure 4 displays a user interface implementing paradigm *e*. According to the canoeing scenario and using this interface, Nicole searches for features of any name that are of type *River* (C_s) and located near Park City, Utah (*1*).



Lake Fork River - Utah, US
Right Hand Fork East Bear River - Utah, US
Little South Fork Provo River - Utah, US
Duchesne River - Utah, US
East Fork Bear River - Utah, US
Left Hand Fork East Fork Bear River - Utah, US
Provo River - Utah, US
Weber River - Utah, US
East Fork Duchesne River - Utah, US
West Fork Duchesne River - Utah, US

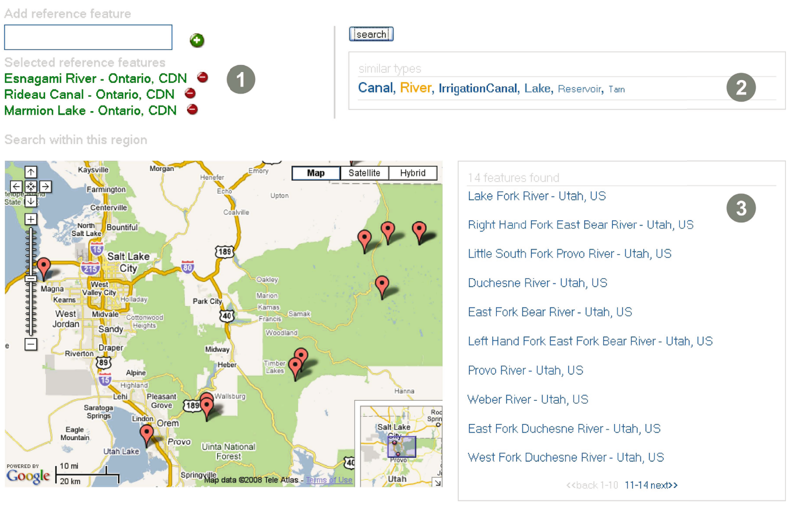
Fig. 4. A conceptual design of a user Web interface illustrating approach *e*

The context concept C_c is defined as LCS of all feature types which have features in the map extent (and is hence set to *Waterbody* with respect to our ontology fragment). As result, a tag cloud showing alternative (similar) feature

[8] Note that implementations for both paradigms exist/are under way for different scenarios: paradigm *e* has been implemented within a similarity-aware gazetteer interface [7]; paradigm *f* is currently being implemented for a similarity-aware climbing route recommendation service. See `http://sim-dl.sourceforge.net/applications/` for details and source codes.

types is returned (*2*). After clicking on the search button, features within the map extent of type *River* are displayed (*3*). The tag cloud can now be used to browse for features that have a different but similar type than *River*.

Figure 5 shows a user interface implementing paradigm *h*. Here, Nicole first specifies known reference features (from Canada) and the map extent (Utah) (*1*). After clicking on the search button, a tag cloud of feature types (C_{t_j}) that are similar to the feature types (C_{r_i}) of the reference features (I_{r_i}) is displayed (*2*). The results are shown for the most similar feature type which is *Canal*, and can be used to display features of other types such as *River* (*3*).



Add reference feature

Selected reference features
Esnagami River - Ontario, CDN
Rideau Canal - Ontario, CDN
Marmion Lake - Ontario, CDN

Search within this region

similar types
Canal, River, IrrigationCanal, Lake, Reservoir, Tarn

14 features found
Lake Fork River - Utah, US
Right Hand Fork East Bear River - Utah, US
Little South Fork Provo River - Utah, US
Duchesne River - Utah, US
East Fork Bear River - Utah, US
Left Hand Fork East Fork Bear River - Utah, US
Provo River - Utah, US
Weber River - Utah, US
East Fork Duchesne River - Utah, US
West Fork Duchesne River - Utah, US
<<back 1-10 11-14 next>>

Fig. 5. A conceptual design for a user Web interface illustrating approach *h*

4.2 SDI Architecture and Workflow

In figure 6, an SDI architecture for implementing similarity-based information retrieval following paradigms *e*, *f*, or *h* is sketched. It assumes a thematic portal as a client application, e.g., a portal serving hydrology information, with a user interface resembling the ones presented in the previous section. The other components in the architecture are standard WMS and WFS instances as well as a catalogue service including a feature type catalogue (CS-W & FTC) and the WSS. We assume that both services as well as the client have access to and use the same ontology. We do not consider here, how the access to the ontology could be enabled through a service interface (as proposed, e.g., in [32]).

The figure shows the workflow for all three paradigms. Until the similarity is computed, the workflow is the same for approaches *f* and *h*, where reference individuals, rather than reference concepts as in *e*, have to be specified. After this step, approaches *e* and *h* share the same workflow as the similarity computation yields concepts rather than individuals as in approach *f*.

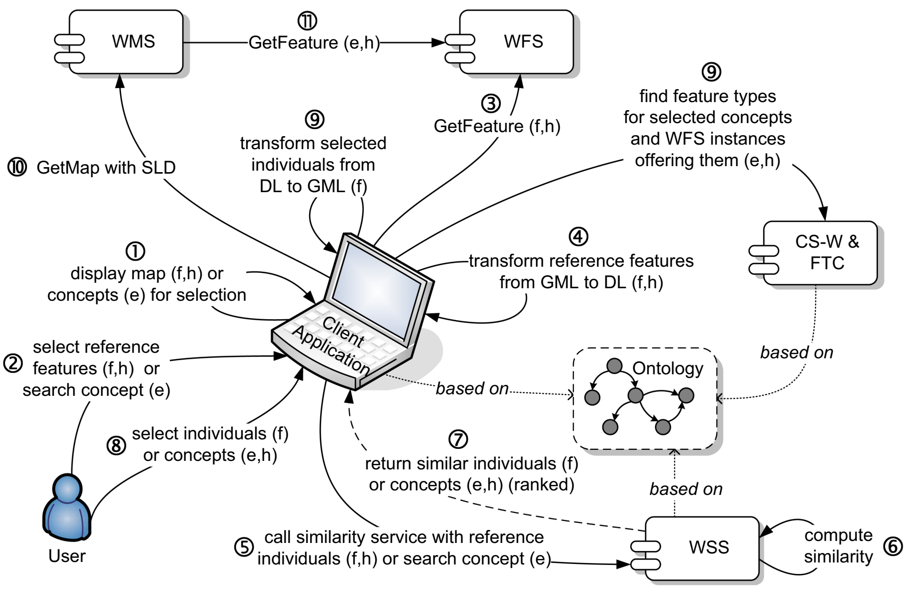

WMS
⑪ GetFeature (e,h)
WFS
⑨ find feature types for selected concepts and WFS instances offering them (e,h)
③ GetFeature (f,h)
⑨ transform selected individuals from DL to GML (f)
⑩ GetMap with SLD
CS-W & FTC
① display map (f,h) or concepts (e) for selection
④ transform reference features from GML to DL (f,h)
Client Application
② select reference features (f,h) or search concept (e)
based on
Ontology
based on
⑧ select individuals (f) or concepts (e,h)
⑦ return similar individuals (f) or concepts (e,h) (ranked)
based on
User
⑤ call similarity service with reference individuals (f,h) or search concept (e)
WSS
compute similarity ⑥

Fig. 6. Architecture and workflow for integrating similarity-based information retrieval into an SDI

In the first step, the client application displays a map using different feature types from the hydrology domain (*f, h*)[9] or a list of possible search concepts from the ontology (approach *e*). The user can then graphically select one or several features in the map (*f, h*) or a concept from the ontology (*2*). Approaches *f* and *h* require that the selected features are retrieved from the WFS (*3*) and translated from their GML representation into DL individuals (*4*). The DL individuals (*f, h*) or concepts (*e*) that are now available in the client are sent to the WSS (*5*), which computes the similarity using the respective approach (*6*). This yields a ranked list of similar individuals (*f*) or concepts (*e, h*) that is displayed in the client (*7*). The user can then select one or several of the presented individuals (*f*) or one of the presented concepts (*e, h*) (*8*). In approach *f*, the selected individuals are translated back into GML, in approaches *e* and *h*, the CS-W is queried for WFS instances offering feature types annotated with the selected concepts (*9*). Based on the GML or WFS instances, the client builds a SLD document and calls a WMS GetMap operation (*10*). The map is created from the GML directly (*f*) or based on a GetFeature requests to the WFS instances listed in the SLD (*e, h*) (11).

4.3 The Missing Pieces

The architecture and workflow sketched in the previous section puts a number of requirements on the used SDI components, in particular the CS-W and WFS.

[9] In an SDI architecture, this will be done using a WMS request, possibly using a styled layer descriptor (SLD) document with references to remote WFS instances. These steps are omitted in figure 6 for readability.

Catalogue Service. The catalogue service needs to store metadata about three types of resources: (1) services, (2) data, and (3) feature types, as well as the relationships between them. The ebRIM catalogue profile for the CS-W allows storing this information in one registry as well as queries combining them. The ebRIM basic extension package describes the relationship between services and datasets through the *OperatesOn* Association defined in ISO 19119 [33]. In [14], an ebRIM extension package is described that allows the storage of feature catalogue (as defined in ISO 19110 [13]) metadata in an ebRIM catalogue. However, there is no association between the feature types and services and/or data. Such an association would be required in order to find WFS instances that provide a specific feature type.

The model defined in ISO 19110 only includes an optional *definition* attribute of type `string` to describe a feature type. In order to do similarity-based search, a link needs to be established to a concept that annotates the feature type. This link should be stored in the (feature) catalogue rather than the ontology in order to avoid having to update the ontology every time a new feature type is registered in the feature catalogue.

Web Feature Service. For the intensional paradigm, the WFS does not have to be changed as the approach works at the feature *type* (rather than the feature *instance*) level and the concepts in the ontology only annotate *features* (rather than also their attributes and/or operations). Thus, once one or several feature types have been discovered, a normal GetFeature request can be sent using the selected feature type (cf. steps 3 and 11 in figure 6).

For the extensional paradigm, features need to be translated into ontology individuals and vice versa (cf. steps 4 and 9). This could be done following the approach described in [34], i.e., by simply mapping the GML properties to DL properties. If the approach is to be successful, this mapping should, wherever possible, use DL roles already existing in the ontology. Otherwise the similarity to existing concepts will be very low. The mapping should preferably be defined by the data provider, or – if no mapping is yet available – by the requester. How to support the creation of such mappings is an open research question.

5 Conclusions and Further Work

This paper investigates paradigms for similarity-based information retrieval, presents prototypical Web user interfaces applying these paradigms, and discusses their integration into SDI as well as remaining difficulties. While the intensional paradigm e has been implemented within SIM-DL and used for a gazetteer research scenario before [7,35], the integration of the new extensional paradigm f within SIM-DL is under development. Further research should especially focus on approach h. It allows to compute inter-concept similarity without requiring the user to define the search and context concept manually. In many application areas, selecting reference individuals, i.e., examples, may be more intuitive both in terms of using the Web user interface as well as in interpreting the results. In several cases, such as the gazetteer scenario [7,35] and to a certain degree also the SDI integration discussed here, geographic features are not available as instances within the ontology; hence paradigm h can be used instead of f (and e) to deliver similar feature types. While the question which of the presented search paradigms (also including those purely based on subsumption reasoning) fits best depends on the application area, it would be fruitful to analyze whether certain scenarios abet a particular paradigm. This could be done by human participants tests, but also by classifying the scenarios. For instance, the benefit of similarity lies in browsing through potential results and reducing the complexity of user interfaces [35], while scenarios which require guaranteed results (e.g., in emergency scenarios) may put more focus on subsumption reasoning. Additionally, the list of subsumption and similarity-based paradigms presented in this paper is not exclusive. On may also think of using logical negation to define the search concept. As pointed out by Nedas and Egenhofer [24], the interpretation of such queries is not trivial in terms of a similarity ranking.

Further research should also focus on how to display the retrieved features and types to the user. While SIM-DL supports value rankings, tag clouds, and categories so far [18], other visualization and interaction methods have to be investigated. For instance (for paradigm f), one may think of sliders to select the similarity threshold value above which features should be displaying on the map.

References

1. Lutz, M., Klien, E.: Ontology-based retrieval of geographic information. International Journal of Geographical Information Science 20(3), 233–260 (2006)
2. Stoimenov, L., Djordjevic-Kajan, S.: An architecture for interoperable gis use in a local community environment. Computers & Geosciences 31, 211–220 (2005)
3. Möller, R., Haarslev, V., Neumann, B.: Semantics-based information retrieval. In: Proc. IT&KNOWS 1998: International Conference on Information Technology and Knowledge Systems, Vienna, Budapest, 31. August- 4. September, pp. 49–56 (1998)
4. Küsters, R.: Non-Standard Inferences in Description Logics. LNCS (LNAI), vol. 2100. Springer, Heidelberg (2001)
5. d'Amato, C., Fanizzi, N., Esposito, F.: A semantic similarity measure for expressive description logics. In: CILC 2005, Convegno Italiano di Logica Computazionale, Rome, Italy (2005)
6. Janowicz, K.: Sim-dl: Towards a semantic similarity measurement theory for the description logic $\mathcal{ALCNR}$ in geographic information retrieval. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1681–1692. Springer, Heidelberg (2006)
7. Janowicz, K., Keßler, C., Schwarz, M., Wilkes, M., Panov, I., Espeter, M., Baeumer, B.: Algorithm, Implementation and Application of the SIM-DL Similarity Server. In: Fonseca, F., Rodríguez, M.A., Levashkin, S. (eds.) GeoS 2007. LNCS, vol. 4853, pp. 128–145. Springer, Heidelberg (2007)
8. Araújo, R., Pinto, H.S.: Towards semantics-based ontology similarity. In: Shvaiko, P., Euzenat, J., Giunchiglia, F., He, B. (eds.) Proceedings of the Workshop on Ontology Matching (OM2007) at ISWC/ASWC2007, Busan, South Korea (2007)
9. Nebert, D.D.: Developing spatial data infrastructures: The SDI cookbook (2004), http://www.gsdi.org/docs2004/Cookbook/cookbookV2.0.pdf

10. McKee, L.: Who wants a GDI?, pp. 13–24. Oxford University Press, Oxford (2000)
11. Bernard, L., Kanellopoulos, I., Annoni, A., Smits, P.: The european geoportal - one step towards the establishment of a european spatial data infrastructure. Computers, Environment and Urban Systems 29, 15–31 (2005)
12. European Commission: Spatial data infrastructures in europe, state of play spring 2005: Summary report of activity 5 of a study commissioned by the ec (eurostat & dgenv) in the framework of the inspire initiative. Technical report, European Commission, Brussels (2005)
13. ISO: ISO 19110:2005 geographic information – methodology for feature cataloguing. International standard, ISO TC 211 (2005)
14. OGC: Feature type catalogue extension package for ebRIM (ISO/TS 15000-3) profile of CSW 2.0. Discussion Paper OGC 07 172r1, Open Geospatial Consortium Inc. (2007)
15. Goldstone, R.L., Son, J.: Similarity. In: Holyoak, K., Morrison, R. (eds.) Cambridge Handbook of Thinking and Reasoning. Cambridge University Press, Cambridge (2005)
16. Medin, D., Goldstone, R., Gentner, D.: Respects for similarity. Psychological Review 100(2), 254–278 (1993)
17. Tversky, A.: Features of similarity. Psychological Review 84(4), 327–352 (1977)
18. Janowicz, K.: Kinds of contexts and their impact on semantic similarity measurement. In: 5th IEEE Workshop on Context Modeling and Reasoning (CoMoRea 2008) at the 6th IEEE International Conference on Pervasive Computing and Communication (PerCom 2008), Hong Kong. IEEE Computer Society Press, Los Alamitos (2008)
19. Keßler, C., Raubal, M., Janowicz, K.: The effect of context on semantic similarity measurement. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2007, Part II. LNCS, vol. 4806, pp. 1274–1284. Springer, Heidelberg (2007)
20. Frank, A.U.: Similarity measures for semantics: What is observed? In: COSIT 2007 Workshop on Semantic Similarity Measurement and Geospatial Applications, Melbourne, Australia (2007)
21. Rodríguez, A., Egenhofer, M.: Comparing geospatial entity classes: an asymmetric and context-dependent similarity measure. International Journal of Geographical Information Science 18(3), 229–256 (2004)
22. Raubal, M.: Formalizing conceptual spaces. In: Varzi, A., Vieu, L. (eds.) Formal Ontology in Information Systems, Proceedings of the Third International Conference (FOIS 2004). Frontiers in Artificial Intelligence and Applications, vol. 114, pp. 153–164. IOS Press, Amsterdam (2004)
23. Li, B., Fonseca, F.: Tdd - a comprehensive model for qualitative spatial similarity assessment. Spatial Cognition and Computation 6(1), 31–62 (2006)
24. Nedas, K., Egenhofer, M.: Spatial similarity queries with logical operators. In: Hadzilacos, T., Manolopoulos, Y., Roddick, J., Theodoridis, Y. (eds.) SSTD 2003. LNCS, vol. 2750, pp. 430–448. Springer, Heidelberg (2003)
25. Gahegan, M., Agrawal, R., Banchuen, T., DiBiase, D.: Building rich, semantic descriptions of learning activities to facilitate reuse in digital libraries. International Journal on Digital Libraries 7(1), 81–97 (2007)
26. Horrocks, I.: Implementation and optimization techniques. In: The description logic handbook: theory, implementation, and applications, pp. 306–346. Cambridge University Press, Cambridge (2003)
27. Wilkes, M., Janowicz, K.: A brief intro to the sim-dl dig extension. Technical report, Institute for Geoinformatics, University of Münster, Germany (2008)

28. Barsalou, L.: Ad hoc categories. Memory & Cognition 11, 211–227 (1983)
29. Barsalou, L.: Intraconcept similarity and its implications for interconcept similarity. In: Similarity and analogical reasoning, pp. 76–121. Cambridge University Press, Cambridge (1989)
30. Janowicz, K., Keßler, C., Panov, I., Wilkes, M., Espeter, M., Schwarz, M.: A study on the cognitive plausibility of SIM-DL similarity rankings for geographic feature types. In: Bernard, L., Friis-Christensen, A., Pundt, H. (eds.) 11th AGILE International Conference on Geographic Information Science (AGILE 2008), Girona, Spain, May 5-8, 2008. Lecture Notes in Geoinformation and Cartography, pp. 115–133. Springer, Heidelberg (2008)
31. Cross, V., Sudkamp, T.: Similarity and Computability in Fuzzy Set Theory: Assessments and Applications. Studies in Fuzziness and Soft Computing, vol. 93. Physica-Verlag (2002)
32. Lacasta, J., Nogueras-Iso, J., Béjar, R., Muro-Medrano, P.R., Zarazaga-Soria, F.J.: A web ontology service to facilitate interoperability within a spatial data infrastructure: Applicability to discovery. Data & Knowledge Engineering 63(3), 945–969 (2007)
33. ISO: ISO 19119:2005 geographic information - services. International standard, ISO TC 211 (2005)
34. Klien, E.: A rule-based strategy for the semantic annotation of geodata. Transactions in GIS, Special Issue on the Geospatial Semantic Web 11(3), 437–452 (2007)
35. Janowicz, K., Keßler, C.: The role of ontology in improving gazetteer interaction. International Journal of Geographical Information Science (IJGIS) 10 (forthcoming, 2008)

Geosensor Data Abstraction for Environmental Monitoring Application

Young Jin Jung and Silvia Nittel

Spatial Information and Engineering, University of Maine, USA
{yjung,nittel}@spatial.maine.edu

Abstract. Environmental observation applications are designed for monitoring phenomena using heterogeneous sensor data types and for providing derived and often integrated information. To effectively handle such a large variety of different sensors, both in scale and type and data volume, we propose a geosensor abstraction for large-scale geosensor networks. Our SGSA(Slope Grid for Sensor Data Abstraction) represents collected data in single grid-based layers, and allows for summarizing the measured data in various integrated grid layers. Within each cell, a slope vector is used to represents the trend of the observed sensor data. This slope is used as a simplifying factor for processing queries over several sensor types. To handle dynamic sensor data, the proposed abstraction model also supports rapid data update by using a mapping table. This model can be utilized as a data representation model in various geosensor network applications.

Keywords: Sensor data abstraction, Geosensor network, Slope grid, GIS, Surface model.

1 Introduction

Environmental monitoring applications have become significant tools for analyzing nature's phenomena. The advances in wireless communication and sensor miniaturization technologies as well as small-form computing devices have significantly contributed to the enablement of environmental monitoring in the physical world [1]. To detect the conditions or events in a wide-area geographic space, a monitoring application requires a large-scale geosensor network [2] including various kinds of sensors. Today, many large, autonomous sensor platforms such as wind sensors or ocean buoys are deployed; in a few years, these traditional sensor environments will be integrated with the deployment of small-form sensor networks providing rapid rates of real-time sensor data of various type, scale and location.

Monitoring applications can provide useful information for environmental science (e.g. a habitat analysis or a micro-climate model for spatially limited area such as a vineyard or a greenhouse) and for a warning system of dangerous environmental events (e.g. forest fire and air pollution) [3]. For developing sensor network applications, various kinds of technologies including sensing, communication and computing are required [4]. Furthermore, the application of domain knowledge is also necessary in order to interpret and understand an environmental condition, based on the collected data [3].

Today, queries in sensor data applications are typically executed on the raw, collected sensor data stored in a database system. To process all sensor data without any data abstraction, query execution slows down significantly due to the large volume of real-time sensor data. In addition, it is difficult to compile useful information for answering queries such as “When will air pollution be reduced?” and “Where is a potential dangerously polluted area in the near future?” It requires a sensor data abstraction as a preprocessing step for interpreting the observed data and for processing a query in a central monitoring server. The sensor data in a wide area can be created in large volume, and streamed to a central server. In addition, it also can change dynamically. In order to handle the observed data, some requirements are posed for the data abstraction model itself, such as a rapid data processing, an effective update policy, and easy data access, etc.

In this paper, we designed SGSA (Slope Grid for Sensor Data Abstraction) for representing large-scale geosensor data in a central monitoring system. The proposed abstraction model is based on a grid-based technique for characterizing terrain surface area [5]. After collecting a sensor data type, the grid represents the data on each cell as a spatial slope that is described by min(), max(), and a slope direction. It can find the condition change in the area because the slope shows the data gradient on the cell. Users can understand the conditions by checking the slopes on cells in the grid as if you would see a simple contour map. Besides, it can support a rapid data update using a mapping table. This proposed model can be utilized as a data representation model in various geosensor network applications.

2 Related Work

The Environment Observation and Forecasting System (EOFS) is an application for understanding a situation and providing forecasting using a large-scale sensor network [6, 7]. The system supports centralized processing, handles huge data volume, and provides an autonomous operation, etc. Some usages examples of EOFS are the habitat monitoring of seabirds [8], the CORIE for monitoring the Columbia river [9], the Automated Local Evaluation in Real-Time(ALERT) [10], and the framework for in-situ sensor data processing [11]. The seabird habitat monitoring project focuses on habitat condition analysis. The goal of the CORIE project is to guide vessel transportation and forecasting. It utilized the 13 stationary sensor nodes and features a computationally intensive physical environment model. GLACSWEB project monitors the behavior of ice caps and glaciers for understanding the Earth’s climate [12]. The in-situ sensor data processing system monitors continuously updated sensor data and the communication status in sensor network [11]. It can provide an alarm with the registered context aware model and rules. The PODS project monitors rare and endangered species of plants by using high-resolution cameras, a thermometer, and solar radiation sensors [13]. It describes a simple data display with colored dots, which shows the category values for each sensor. For example, green dots mean a normal condition. On the other hand, red indicates one of the extreme categories. It also considers data summarization techniques such as consolidating the data with Theissen polygons (Delaunay triangulation) [14] to show the general pattern of the data values as a map. GMT, the Generic Mapping tools, is available to present the measured data [15].

On the spatial resolution, some applications need the raw data of all sensing points [16, 17], whereas others need just a summary of all sensors' data, as those TAG [18]. An intermediate data summarization between these two cases is also utilized such as maps of temperature and relative humidity [7]. These applications can identify zones of interest such as hot and cold zones. To avoid information overload, spatial aggregation is also utilized for data summarization over subregions, pre-defined zones. For example, it uses a aggregation predicate such as "SELECT avg(volume) FROM Sensors GROUP BY region HAVING avg(volume) > threshold." [19] The spatial distribution of these summarizations provides a report of the data variability over the entire region. In order to recognize the conditions of the circumstance, monitoring applications including EOFS need a well-organized data representation model, because they have to rapidly process a huge volume of sensor data and interpret the transmitted observed data.

3 Slope Grid for Sensor Data Abstraction

In a large-scale geosensor network, sensor data abstraction is required for handling large volumes of data and making the information useful even though a large scale sensor network has not yet been utilized for practical environmental monitoring until recently [7]. The volume of sensor data can potentially be generated over a large geographic area, and we assume that the collected values dynamically change continuously. A grid is generally utilized to present the measured data or the condition on an area such as a graph and a terrain surface model in GIS [20]. Our slope grid is based on the tilted plane, which is useful to present a surface area [5]. When it present a surface area, a slope (the tilted plane) is better than the horizontal plane, because the slope can present the data elevation change as well as the height in each cell. When it is used to manage a large volume of moving object location data, this grid structure also shows good search performance by using a hash table as an index structure [21]. It focuses on the access method for updating and searching the location of moving objects. In this paper, we concentrate the summarization of environmental condition for analyzing a phenomenon with slope grid.

If an application stores the measured sensor data without any data spatio-temporal abstraction model, all data has to be searched for answering user queries. It requires more time to process the queries and to derive the information that users want to know. The objectives of providing a sensor data abstraction model are the frequent data updates, the geographic area covered, and a continuous amount data process, data representation and a compact size, etc.

To apply the slope for surface areas to the sensor data monitoring application, we simplify the presentation of the tilted plane to 9 directions derived from min() and a max() of 4 subcells in a cell. In order to present the continuously changed data motion, we use an update policy with a mapping table and a gradient count for checking the data change in a cell.

We propose sensor data abstraction model, in which sensor data is represented in individual grids. The grid for each type serves as input for spatio-temporal sensor queries, which are sophisticated data interpretation in most cases. In our approach, we use the data abstraction model as a preprocessing step for interpreting the conditions at the remote place, thus, as a basic data representation model for an environmental monitoring application [11]. It focuses only on current data. This abstract data is used for making useful information by combining other abstracted data types.

The proposed sensor data abstraction is shown in Figure 1. The data is summarized on each cell after receiving the sensor data transmitted from geosensor network at (a). We choose the minimum and the maximum value for representing values within a cell at Figure 2(b). The other values of the cell are included in the value boundary, which consists of min() and max(). The size of the cell is defined depending on some conditions such as the number of sensors in a cell, data feature, and application function. In a cell, 1~8 sensors are useful to present the trend of data, because a cell includes 4 subcells which present a min() and a max() with two data values within each subcell. It can include more than 8 observed values but the data representation is not different. The size is also defined by a data feature. For example, it does not require many sensors for observing wind direction and speed, because both phenomena don't frequently changed over a larger area. In this paper, we consider only a static cell size since changing a cell area also requires the time to process it. It would be an obstacle for the rapid data processing.

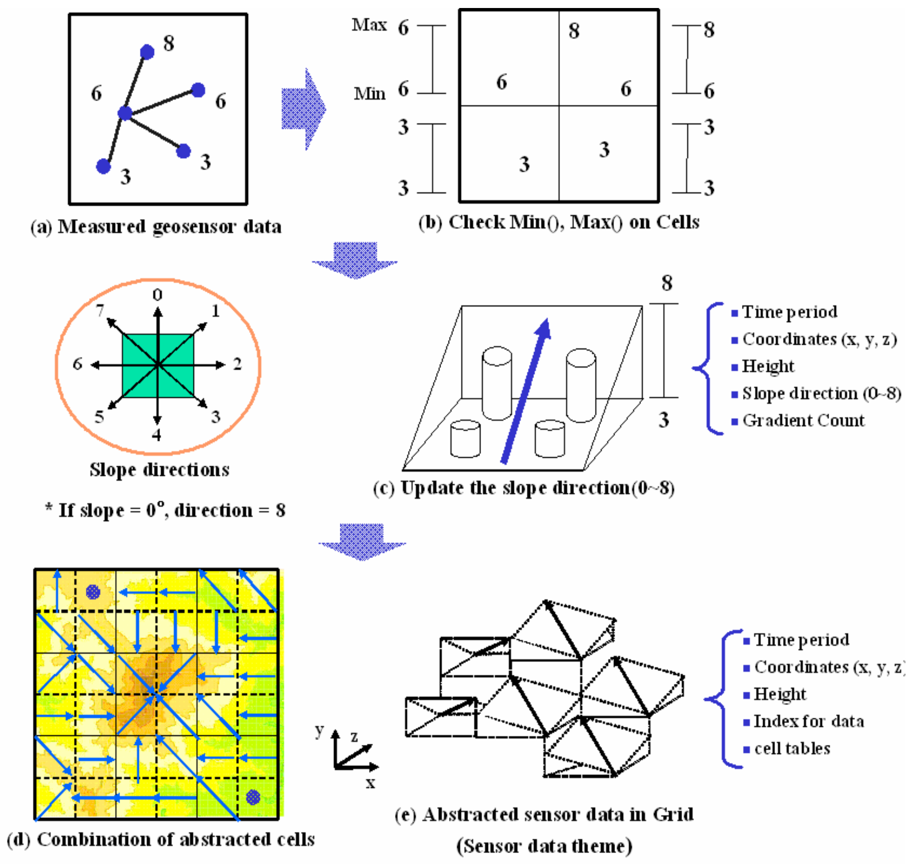
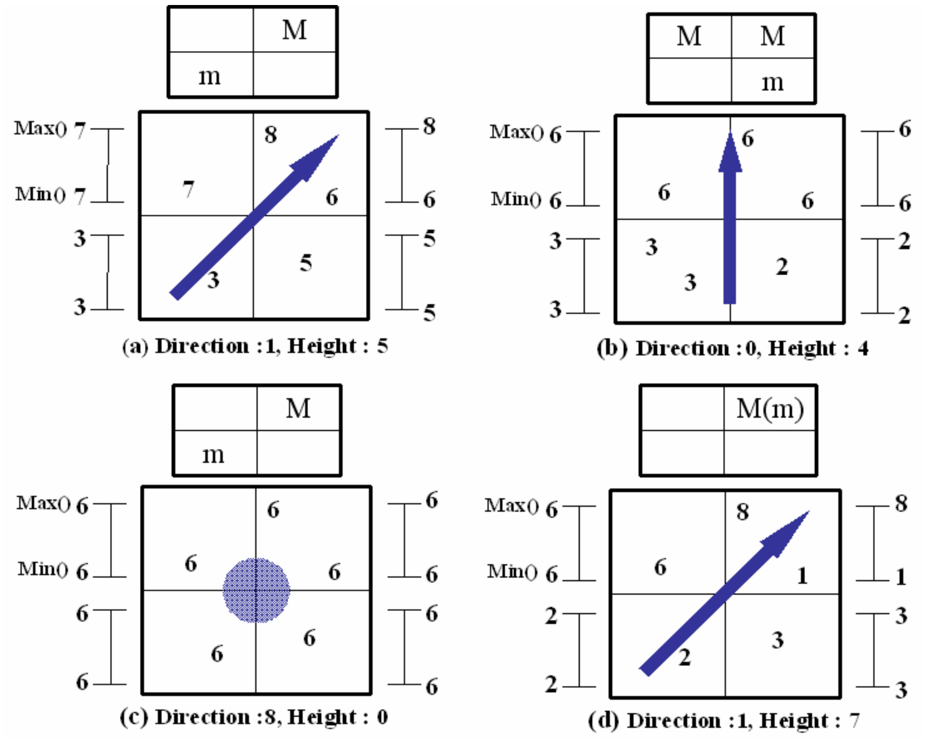


Fig. 1. The sensor data abstraction

The coordinates (x_1, y_1, z_1, x_2, y_2, z_2) indicate the locations of cells in a grid. The time period (t_1, t_2) of a cell includes all of the observation time of the sensor data at this location. It contains the latest measurement time, because it is updated whenever the observed data is updated. At (c), the height is a difference between min() and max() in the cell. If max() is 8 and min() is 3, the height is 5 and z coordinate is 3, which starts at the min(). The slope direction is derived from the relative positions of two subcells as shown in Figure 2. One is the "M" subcell, which contains the maximum value, and the other is the "m" subcell which has a minimum value.



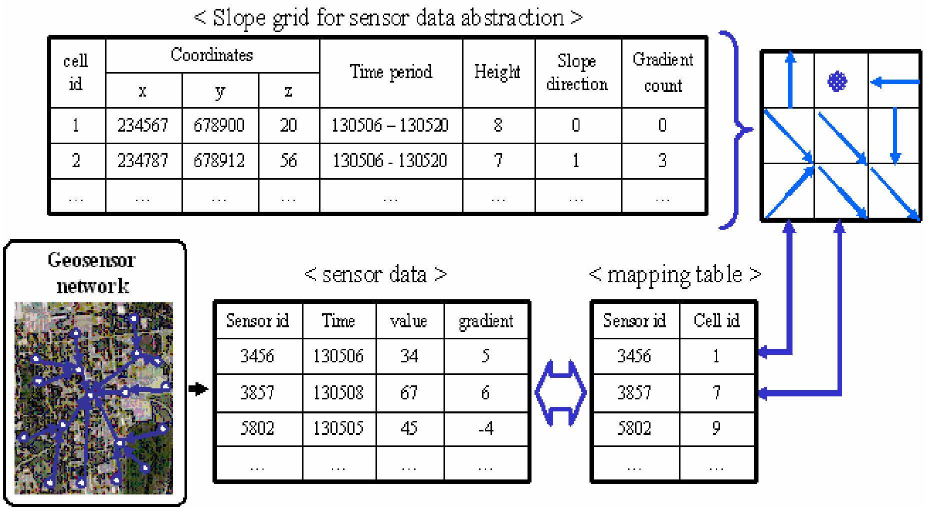
M
m
Max() 7
Min() 7
8
7
6
3
5
(a) Direction :1, Height : 5
M
M
m
Max() 6
Min() 6
6
3
2
(b) Direction :0, Height : 4
M
m
Max() 6
Min() 6
6
(c) Direction :8, Height : 0
M(m)
Max() 6
Min() 6
8
6
1
2
3
(d) Direction :1, Height : 7

Fig. 2. The derived slope direction

The direction vector is a vector pointing from m subcell to M subcell. Namely, it presents the direction from the min() to the max() in a cell (Figure 2). In a small grid above each cell, M indicates the M subcell and m the m subcell. If the slope angle = 0°, the direction = 8. It means that all the sensor data has the same value and the slope is flat. It is presented a dot in the grid at (c) in Figure 2. If a subcell includes a maximum and a minimum values like (d), the direction points towards the M subcell. Namely, the direction always points the M subcell. When it present the data in a cell, this slope is better than a horizontal plane, because the slope present the data trend. For example, if there are two horizontal planes, which have same height, they are only equal. However, if there are two slopes with same height, they can present various kinds of data motion even though they are same height.

4 Updating Data to the Abstract Model

It is essential to support frequent data update in sensor data monitoring applications, especially for real-time monitoring and warning systems. Such a system has to detect and cope with the observed emergency situation such as air pollution, forest fire, battlefield analysis, etc. Sensor data likely changes dynamically over time. The abstracted data representation model employs a mapping table for rapid data update.



cell id	Coordinates			Time period	Height	Slope direction	Gradient count
	x	y	z				
1	234567	678900	20	130506 – 130520	8	0	0
2	234787	678912	56	130506 - 130520	7	1	3
…	…	…	…	…	…	…	…

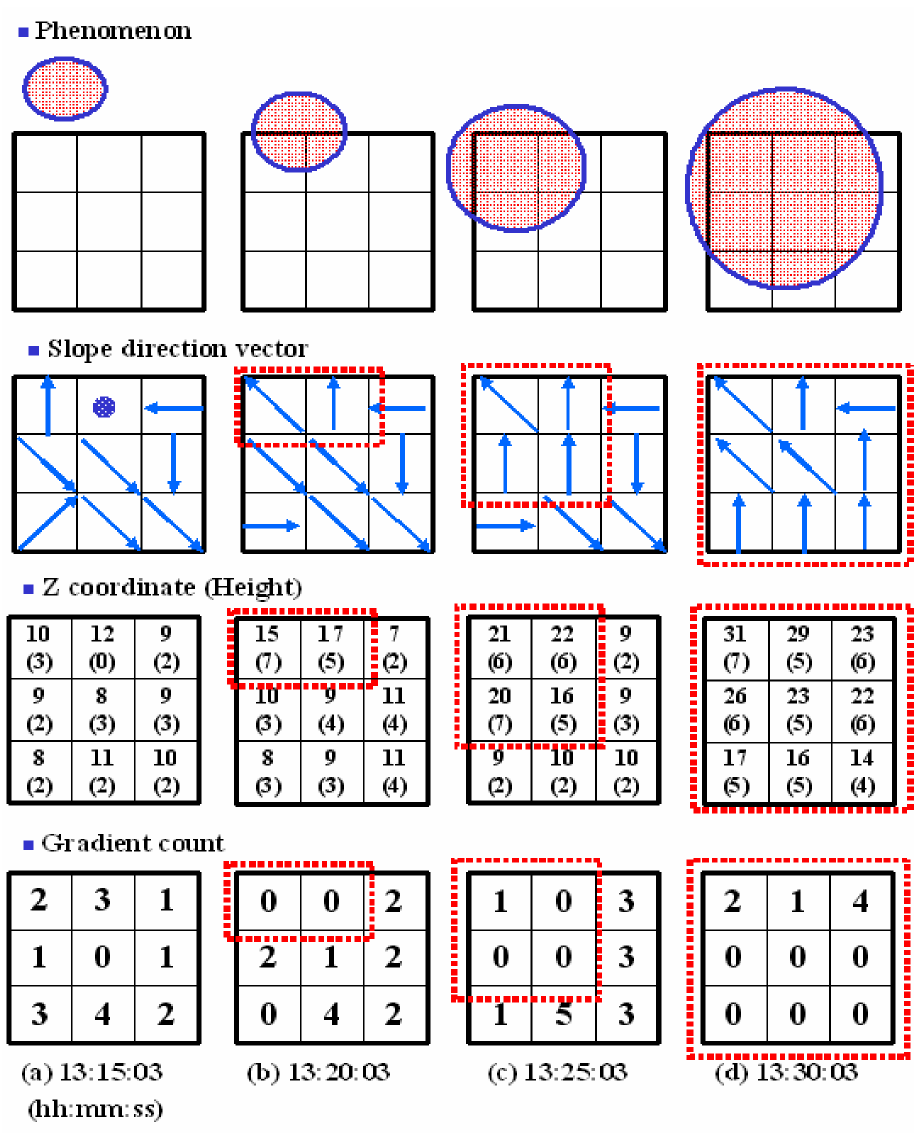
Sensor id	Time	value	gradient
3456	130506	34	5
3857	130508	67	6
5802	130505	45	-4
…	…	…	…

Sensor id	Cell id
3456	1
3857	7
5802	9
…	…

Fig. 3. The sensor data update structure in the SGSA

The update process of the abstract data model is shown in Figure 3. The transmitted data is stored in an incoming sensor data table or data stream. The sensor data table stores the sensor id, the measurement time stamp, the sensed value, and the gradient. The gradient shows data variation derived from past value. The mapping table shows the cell id and included sensor id. Whenever the locations of sensors are changed, it finds the cells, which include the locations. This data is summarized in the abstracted data table for each cell. Here, the slope direction and the gradient count are also updated by changing the minimum and maximum values. However, if there is already a lower minimum value or higher maximum value, the values are not changed. So, it is designed to update the values only whenever the time period or the slope direction is changed. The updatable area in the SGSA can be limited by defining the boundary of a specific area depending on applications.

Figure 4 shows the changed representation of the SGSA depending on an observed phenomenon. For example, we can imagine a temperature change on a fire area. Users of environmental monitoring applications can understand and cope with the event by analyzing these changed values. When a geosensor network detects a phenomenon, it changes the attributes in SGSA such as slope direction, z coordinate, height, and gradient count. First, the z coordinate and the height are changed by updating the sensed data. Next, the slope direction is derived from the m and M subcells in each cell.



Z coordinate (Height)

(a) 13:15:03

10 (3)	12 (0)	9 (2)
9 (2)	8 (3)	9 (3)
8 (2)	11 (2)	10 (2)

(b) 13:20:03

15 (7)	17 (5)	7 (2)
10 (3)	9 (4)	11 (4)
8 (3)	9 (3)	11 (4)

(c) 13:25:03

21 (6)	22 (6)	9 (2)
20 (7)	16 (5)	9 (3)
9 (2)	10 (2)	10 (2)

(d) 13:30:03

31 (7)	29 (5)	23 (6)
26 (6)	23 (5)	22 (6)
17 (5)	16 (5)	14 (4)

Gradient count

(a) 13:15:03

2	3	1
1	0	1
3	4	2

(b) 13:20:03

0	0	2
2	1	2
0	4	2

(c) 13:25:03

1	0	3
0	0	3
1	5	3

(d) 13:30:03

2	1	4
0	0	0
0	0	0

Fig. 4. The representation update of the SGSA

Users can recognize the trend change of the observed data by searching the direction and the height, because the direction points are higher. Finally, the gradient count is updated depending on the direction change. The gradient count shows how long the value of the direction has been kept. Whenever the sensor data is updated, if the direction is not changed, it increases the count. In other words, a high count means the stable condition in the cell. If the direction is changed, it sets the count to 0. It means that the condition in the cell is changed. If the count is almost 0 in a long time, it indicates the fact that the condition of the cell, or the area, is frequently changed. This count is a useful factor for evaluating the dynamically changing condition of an area. Users can find some areas (cells), which shows recently changed data motion by searching low gradient count. These areas are utilized for finding a boundary of a phenomenon or recognizing areas which show frequently changed data motion. These attributes are used for tracking and analyzing a phenomenon.

5 Utilization in Sensor Data Query Processing

In sensor data monitoring applications, users are interested in posing higher-level queries to understand the information collected via sensors in a situation in a specific area. For example, a habitat monitoring system analyzes different conditions among groups of animals or an environmental monitoring application tracks the interesting phenomena such as red tide, a forest fire, and air pollution.

The abstract data types are used as the basis for answering a query, because of its summarized data representation on spatial area by describing a data gradient, min(), max(), and gradient count. An example of query processing on the abstracted data is shown in Figure 5. Assume, a user asks the query "find all wildfire fires in the

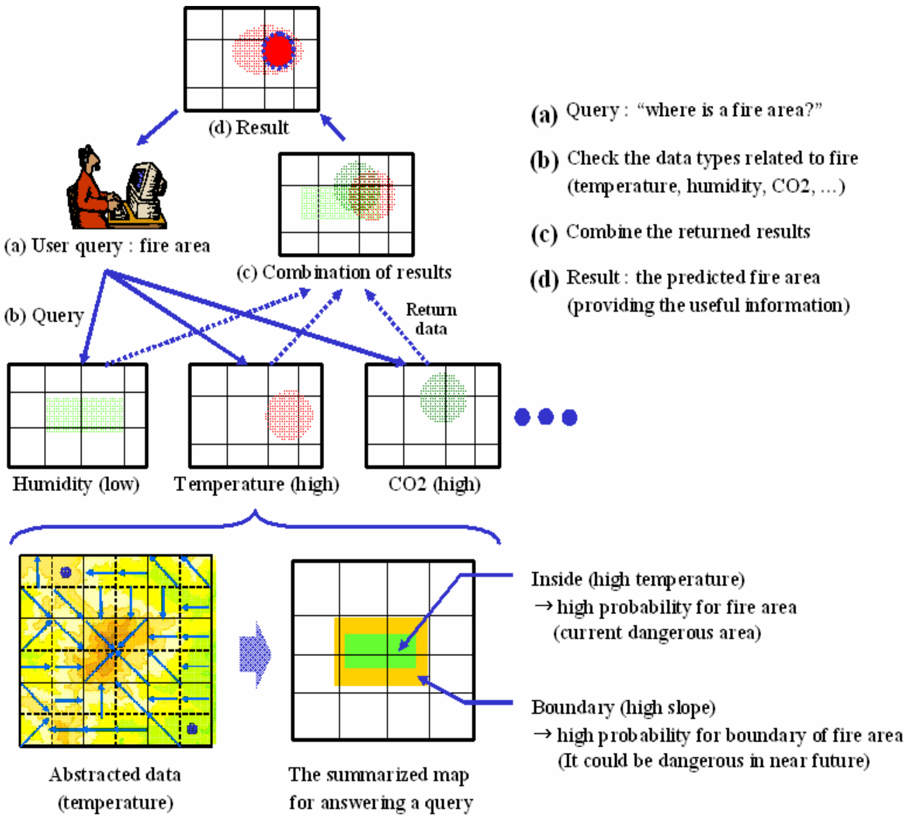

(d) Result
(a) User query : fire area
(c) Combination of results
(b) Query
Return data
Humidity (low)
Temperature (high)
CO2 (high)
(a) Query : "where is a fire area?"
(b) Check the data types related to fire (temperature, humidity, CO2, ...)
(c) Combine the returned results
(d) Result : the predicted fire area (providing the useful information)
Abstracted data (temperature)
The summarized map for answering a query
Inside (high temperature) → high probability for fire area (current dangerous area)
Boundary (high slope) → high probability for boundary of fire area (It could be dangerous in near future)

Fig. 5. The update of the abstracted data in the Grid

area" at (a). Once the query is parsed, the monitoring system registers it as a continuous query. In order to answer the query, the system creates subqueries to analyze the related abstracted data within the observation area at (b). For example, two kinds of subqueries are issued to the temperature theme (abstract data) such as "find the area(s) detecting high temperature" and "find the area(s) detecting a high slope." The result of the first subquery can be a factor for finding a currently burning area or a burned area, because it is already hot. The result of the second subquery can be an indicator for finding areas, which could be burnt in near future by checking the high slope and the gradient count. The high slope and low gradient count mean that the data on the cell is rapidly changed. The cell detecting a high slop or a low gradient count could be a boundary of fire area. During this query process, if it requires a detail data on a cell, it can also search the sensor data by using the mapping table in Figure 3. At (c), the results can be combined after these two kinds of results are returned from several abstract data layers. Some information is extracted from the combined area depending on the predefined information about a fire area. It then can provide the users with a probability of an event at (d).

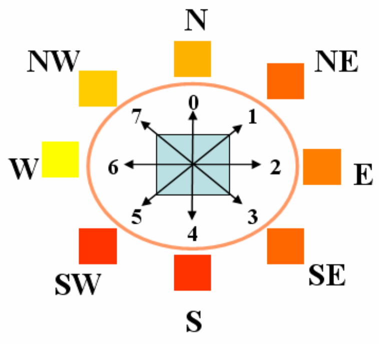
The results of the query will be continuously updated, since most of queries are continuous queries. Therefore, whenever sensor data is updated, the query result needs to be updated. These steps show the example of query processing with the abstracted model. It should predefine the relation among phenomenon (e.g. fire) and data types before the query processing. It could include some kinds of rules for combining several data types.

6 Implementation

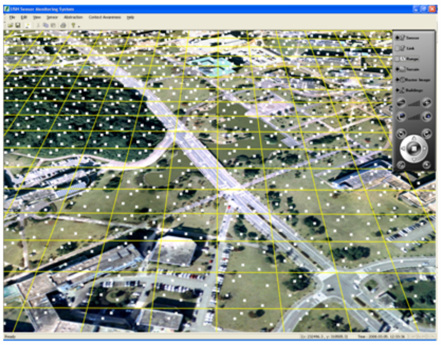
The designed slope grid is implemented with 60,000 simulated static sensors and 10,000 cells as shown in (b) of Figure 6. It assumes that all of the observed data is transmitted to the central processing server. A data generator is used for creating simulated sensor data. Whenever the sensor data is updated by the generator, it updates the attributes of the cells in the grid such as max(), min(), and the slope direction. The slopes and gradient counts of cells are derived from the attribute values in each cell. Finally, it makes a summarized 3 dimensional map with SGSA for presenting the conditions of a remote place. This map is changed over time depending on the observed data change.

Two kinds of data types are used for testing the abstraction model such as random data and event data. When it generates all of random sensor data over time, the slopes are continuously changed like a wave at (c). Event data is used to detect an accident such as a fire, pollution, and red tide areas. The simulated event data is generated according to the parameters of sensor data generator such as height, width, and maximum (or minimum) value. The attributes of the cells are also updated according to the abstraction model. There are various kinds of data representation depending on the data change such as a variety of natural geographical features. In (d) event data of Figure 6 shows rapidly changed slopes in a specific area like a fire area. The maximum value and boundary of the event is used to process user queries as shown in Figure 5. In order to get the useful data in SGSA, some rules are used for extracting the specific cells having specific conditions. For example, to get the cells having the high value over mean(), it uses a rule that is min() in cells > mean(). To get the boundaries of a phenomenon, it also uses rules that are height > 20 and gradient count < 5, because high height and low gradient count mean rapidly changed condition in a cell.

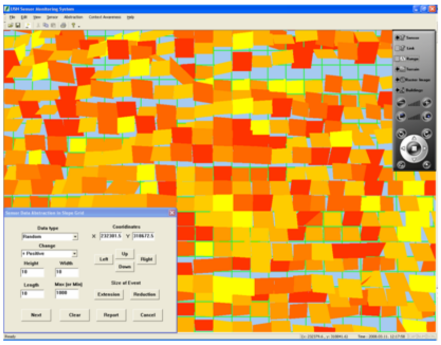
Figure 7 shows the tracking phenomenon which is simulated by data generator. The generator makes the simulated data with predefined conditions such as height (10~30), radius (40 m). The created circle having high values is moved over time according to the user defined route. When the circle is detected by sensors at t1, it makes a map of data with SGSA. The map is changed depending on the simulated circle's movement. We can extract the boundaries of the circle by simple rules for finding the cells which have the height over 17 and the gradient count less than 2. The red lines indicate the current detected boundaries on the left side pictures. The dot blue lines presented past boundaries. On the right side pictures, the boundaries are shown by SGSA. From these abstracted data, we can extract the information of the detected phenomenon such as the speed of movement, the boundaries, and the height of data values. It is useful to answer these questions such as "How fast does the phenomenon travel?", "What is the data gradient average in its boundaries?" To support a complex query, it is required to combine the information extracted from the several SGSAs for various data types.



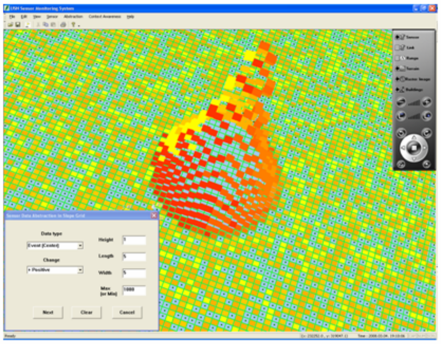
(a) Slope Direction and Color



(b) Sensors in the SGSA



(c) Random Data in the SGSA



(d) Event Data in the SGSA

Fig. 6. Data representation in SGSA

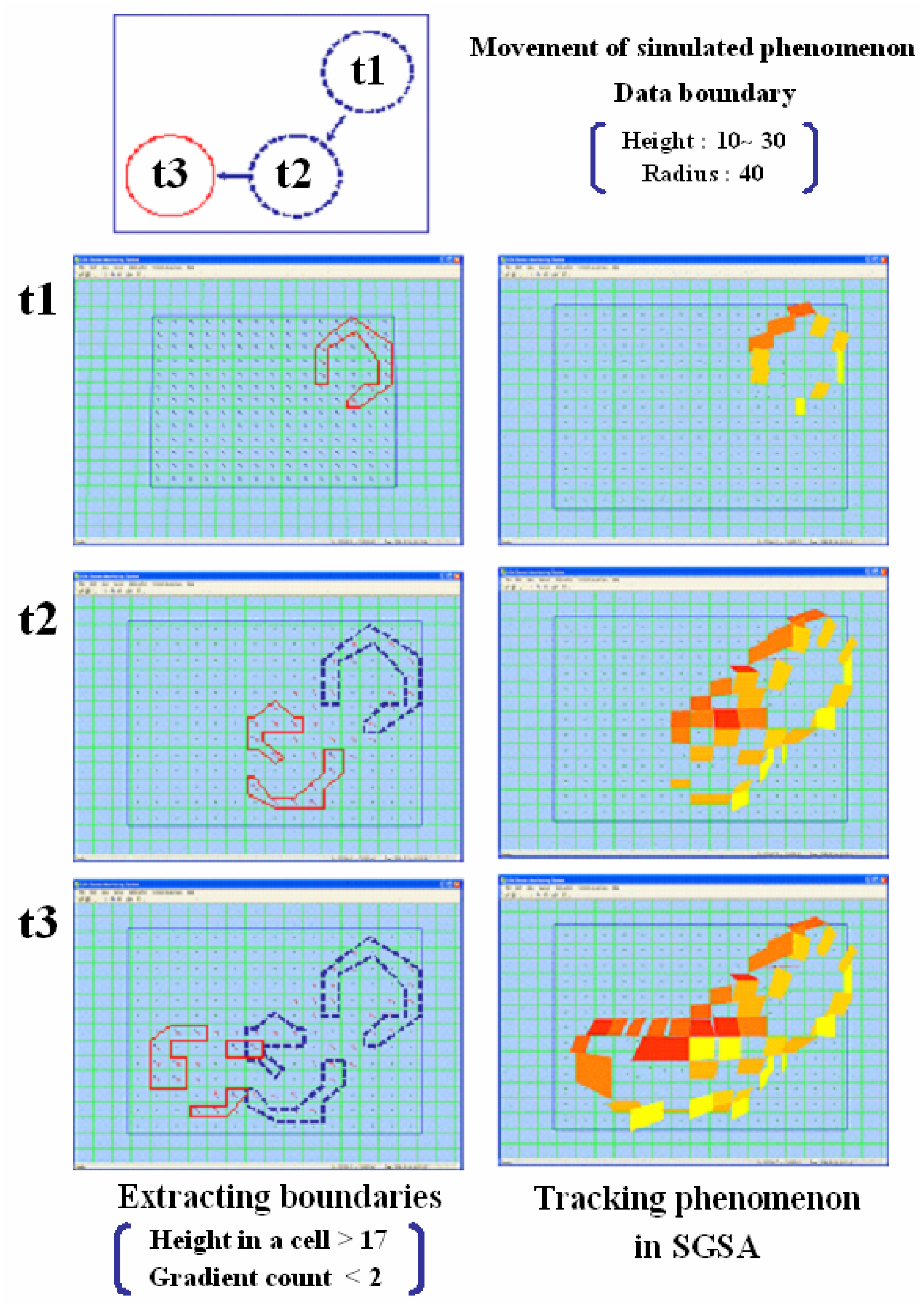


Fig. 7. Tracking the simulated phenomenon with SGSA

7 Conclusions

Environmental monitoring applications deal with the data sets of different types as well as merge them to enhance our understanding of the circumstance. It is required to abstract the sensor data as a preprocessing step for interpreting the condition on a specific area. We designed the data representation abstraction for large-scale geosensor data in a central monitoring system. The slopes on cells represent the observed data in grid like a summarized contour map. It also stores a gradient count and a slope direction for finding the condition change in a wide area. It could be utilized as a data representation model in the environmental monitoring applications. In the future, this model can be extended by using the combination method among the heterogeneous abstracted data types for a sensor data fusion.

Acknowledgement

This work was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD). No. KRF-2007-357-D00206

References

1. Elson, J., Estrin, D.: Sensor networks: a bridge to the physical world. Wireless Sensor Networks, 3–20 (2004)
2. Nittel, S., Stefanidis, A.: GeoSensor Networks and Virtual GeoReality. GeoSensors Networks 296 (2005)
3. Martinez, K., Hart, J.K., Ong, R.: Environmental Sensor Networks. IEEE Computer 37(8), 50–56 (2004)
4. Chong, C.Y., Kumar, S.P.: Sensor Networks: Evolution, Opportunities, and Challenges. Proceedings of the IEEE 91(8), 1247–1256 (2003)
5. Lopez, M., Berry, J.K.: Use Surface Area for Realistic Calculations. GeoWorld, pp. 22-23 (2002)
6. Xu, N.: A Survey of Sensor Network Applications. IEEE Communications Magazine 40(8), 102–114 (2002)
7. Ilka, A.R., Gilberto, C., Renato, A., Antônio, M.V.M.: Data-Aware Clustering for Geosensor Networks Data Collection. In: Anais XIII Simpósio Brasileiro de Sensoriamento Remoto, INPE, pp. 6059–6066 (2007)
8. Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., Anderson, J.: Wireless Sensor Networks for Habitat Monitoring. In: ACM International Workshop on Wireless Sensor Networks and Applications, EUA, pp. 88–97 (2002)
9. Milburn, H.B., Makamua, A.I., Gonzalez, F.I.: Real-Time Tsunami Reporting from the Deep Ocean. In: OCEANS 1996, MTS/IEEE conference, vol. 1, pp. 390–394 (1996)
10. ALERT, http://www.alertsystems.org
11. Jung, Y.J., Lee, Y.K., Lee, D.G., Park, M., Ryu, K.H., Kim, H.C., Kim, K.O.: A Framework of In-situ Sensor Data Processing System for Context Awareness. In: ICIC, pp. 124-129 (2006)

12. Biagioni, E., Bridges, K.: The application of remote sensor technology to assist the recovery of rare and endangered species. Special issue on Distributed Sensor Networks for the International Journal of High Performance Computing Applications 16(3) (2002)
13. Hart, J.K., Rose, J.: Approaches to the study of glacier bed deformation. Quaternary International 86, 45–58 (2001)
14. Delaunay, B.: Sur la sphère vide, Izvestia Akademii Nauk SSSR. Otdelenie Matematicheskikh i Estestvennykh Nauk 7, 793–800 (1934)
15. Generic Mapping tools, http://gmt.soest.hawaii.edu
16. Chu, D., Deshpande, A., Hellerstein, J.M., Hong, W.: Approximate Data Collection in Sensor Networks using Probabilistic Models. In: International Conference on Data Engineering, p. 48 (2006)
17. Tulone, D., Madden, S.: PAQ: Time Series Forecasting For Approximate Query Answering In Sensor Networks. In: Römer, K., Karl, H., Mattern, F. (eds.) EWSN 2006. LNCS, vol. 3868, pp. 21–37. Springer, Heidelberg (2006)
18. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tag: a tiny aggregation service for ad-hoc sensor networks. SIGOPS Operating Systems Review, 31–46 (2002)
19. Goldin, D.: Faster In-Network Evaluation of Spatial Aggregation in Sensor Networks. In: Int'l IEEE Conference On Data Engineering, p. 148 (2006)
20. Rigaux, P., Scholl, M., Voisard, A.: Spatial Databases with application to GIS. Morgan Kaufmann Publishers, San Francisco (2002)
21. Choi, W.: Adaptive cell-based index for moving objects. Data & Knowledge Engineering archive 48(1), 75–101 (2004)

The 9^{+}-Intersection: A Universal Framework for Modeling Topological Relations

Yohei Kurata

SFB/TR8 Spatial Cognition, Universität Bremen
Postfach 330 440, 28334 Bremen, Germany
ykurata@informatik.uni-bremen.de

Abstract. The 9^{+}-intersection is an extension of the 9-intersection, which distinguishes the topological relations between various spatial objects by the pattern of a nested matrix. This paper develops a small set of constraints on this matrix, which is applicable to arbitrary pairs of spatial objects in various spaces. Based on this set of universal constraints, the sets of matrix patterns, each representing a candidate for topological relations, are derived for every possible pair of basic objects (points, directed/non-directed line segments, regions, and bodies) embedded in $\mathbf{R}^1$, $\mathbf{R}^2$, $\mathbf{R}^3$, $\mathbf{S}^1$, and $\mathbf{S}^2$. The derived sets of candidates are consistent with the sets of topological relations ever identified, as well as yield the identification of some missing sets of topological relations. Finally, the topological relations between a region and a region with a hole in $\mathbf{R}^2$ and $\mathbf{S}^2$ are identified to demonstrate the applicability of our approach to deriving topological relations between more complicated objects.

1 Introduction

Topological relations between two spatial objects, which concern how the objects intersect with each other, have been studied extensively in the geographic database community, motivated by the necessity of a formal and cognitively-adequate basis of spatial query language. Previous studies have identified sets of all possible topological relations between various pairs of objects in $\mathbf{R}^2$ [1-7], as well as in $\mathbf{R}^1$ [8], $\mathbf{R}^3$ [9], $\mathbf{S}^1$ [10], $\mathbf{S}^2$ [11], and $\mathbf{Z}^2$ [12]. $\mathbf{R}^n$ is an n-dimensional Euclidean space, $\mathbf{S}^1$ is a circle (1-sphere), and $\mathbf{S}^2$ is an ordinary sphere (2-sphere), and $\mathbf{Z}^2$ is a discrete raster space. Each identified set of topological relations has a specific practical value. For instance, topological line-region relations in $\mathbf{R}^2$ are useful for modeling spatial predicates related to motions, such as *enter* and *go across* [13], and topological line-line relations in $\mathbf{R}^1$ or $\mathbf{S}^1$ are useful for modeling temporal relations [8, 10].

As a formal model of topological relations, many studies have adopted the 4-intersection [1], the 9-intersection [2], or their extension [5, 6, 10, 14]. In these models, topological relations between two objects A and B are represented by the patterns of a matrix, whose elements represent the intersections between point sets associated with A and those associated with B. Usually, candidates for possible topological relations between A and B are derived computationally as the set of matrix patterns that satisfy certain constraints. The set of candidates, each with at least one geometric realization, is approved as the set of all possible topological relations between A and B. Previous studies developed such a constraint set on the matrix for each pair of objects in each space [1, 2, 4-7, 9-12]. Although there are certain overlaps between such constraint sets [2, 9], it is a hard step to develop a constraint set in order to study a new set of topological relations. As a solution to this problem, this paper develops a set of *universal constraints*, which is applicable to arbitrary pairs of objects embedded in various spaces.

Our universal constraints are applied to the matrix of the 9^+-intersection [6]. The 9^+-intersection is an extension of the 9-intersection, which supports the subdivision of objects' interior, boundary, and exterior. For instance, the exterior of a region with a hole is subdivided into outer and inner exterior subsets. Thanks to the support of such subdivisions, the 9^+-intersection is able to capture the topological relations between various spatial objects, including complicated ones. Accordingly, making use of the 9^+-intersection together with the set of universal constraints, we can easily derive the candidates for the topological relations between various pair of objects embedded in various spaces. Indeed, this paper derives the sets of such candidates for every possible pair of basic objects (points, directed/non-directed lines, regions, and bodies) embedded in $\mathbf{R}^1$, $\mathbf{R}^2$, $\mathbf{R}^3$, $\mathbf{S}^1$, and $\mathbf{S}^2$. In this paper, lines and regions normally refer to *simple lines* and *simple regions* [4, 15], respectively. Simple lines are lines with no self-intersection and no branch, derived by a one-to-one mapping from [0,1] to the space. If their two endpoints are ordered, lines are called *DLines* (directed lines). Simple regions are two-dimensional point sets with a connected interior, no hole, and no spike, as well as no fin in $\mathbf{R}^3$. Finally, bodies refer to *simple bodies*, which are three-dimensional counterparts of simple regions.

The remainder of this paper is structured as follows: Section 2 introduces the 9^+-intersection and its matrix-based representation. Section 3 develops a set of universal constraints on the matrix of the 9^+-intersection. Based on this set of constraints, Section 4 derives the candidates for topological relations between every pair of basic objects and analyzes the candidates in comparison with the topological relations identified in the previous studies. Section 5 derives further candidates of basic topological relations by converting the matrix patterns derived in Section 4 and analyzes these new candidates. Section 6 demonstrates the applicability of our approach to the derivation of topological relations between more complicated objects. Finally, Section 7 concludes with the discussion of a future problem.

2 The 9^+-Intersection

The 9-intersection [2] is a model of topological relations between two spatial objects. Based on point-set topology [16], this model distinguishes the *interior*, *boundary*, and *exterior* of each object, which are also called the object's *topological parts*. Let X be a spatial object and $\overline{X}$ be X's *closure* (the intersection of all closed point sets that contain X). Uppercase letters are used because spatial objects are considered sets of points. X's interior X° is the union of all open sets contained in X, X's boundary ∂X is the difference between $\overline{X}$ and X°, and X's exterior X^- is $\overline{X}$'s complement. Accordingly, the boundary of a region refers to its looped edge, the boundary of a line refers to its two endpoints, and the boundary of a point refers to the point itself.

The 9-intersection captures the topological relation between two spatial objects *A* and *B* based on the intersections of *A*'s three topological parts and B's three topological parts. These 3 × 3 = 9 types of intersections are concisely represented by the *9-intersection matrix* in Eqn. 1. Normally, topological relations are distinguished by the presence or absence of these nine types of intersections. The use of some additional properties of the intersections are also proposed for more detailed distinction of topological relations [14, 17].

$$M(A,B)=\begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} \tag{1}$$

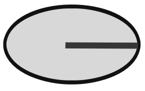
Each topological part of a spatial object may be subdivided into multiple subparts based on their disconnection or qualitative difference (e.g., dimensions [18]). For instance, the boundary of a DLine is subdivided into two subparts; start-point and end-point [5]. In order to support such subdivision of objects' topological parts, the *9$^+$-intersection* [6] extends the 9-intersection, considering the intersections between the subparts of *A*'s three topological parts and those of *B*'s three topological parts. In this model, the topological relations between *A* and *B* are characterized by the *9$^+$-intersection matrix* in Eqn. 2, whose nine bracketed elements are matrices by themselves, each representing the intersections between the subparts of *A*'s one topological part and those of *B*'s one topological part. A°_i}, $\partial_i A$, and A^{-_i} are the *i*th subpart of *A*'s interior, boundary, and exterior, while B°_j}, $\partial_j B$, and B^{-_j} are the *j*th subpart of *B*'s interior, boundary, and exterior, respectively. If a topological part is not subdivided, we consider that this topological part consists of a single subpart. As seen from the comparison of the matrices in Eqns. 1-2, the 9$^+$-intersection matrix keeps the framework of the 9-intersection matrix; that is, the nine inner matrices in the 9$^+$-intersection matrix uniquely correspond to the nine elements in the 9-intersection matrix. Just like the 9-intersection, topological relations between *A* and *B* are distinguished by the presence or absence of all intersections listed in the matrix.

$$M^+(A,B)=\begin{pmatrix} \left[A^{\circ_i} \cap B^{\circ_j}\right] & \left[A^{\circ_i} \cap \partial_j B\right] & \left[A^{\circ_i} \cap B^{-_j}\right] \\ \left[\partial_i A \cap B^{\circ_j}\right] & \left[\partial_i A \cap \partial_j B\right] & \left[\partial_i A \cap B^{-_j}\right] \\ \left[A^{-_i} \cap B^{\circ_j}\right] & \left[A^{-_i} \cap \partial_j B\right] & \left[A^{-_i} \cap B^{-_j}\right] \end{pmatrix} \tag{2}$$

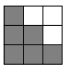
As an example, Eqn. 3 shows the 9$^+$-intersection matrix for the topological relation between a DLine *D* and a region *R*. In this matrix, $\partial_1 D$ and $\partial_2 D$ represent *D*'s start-point and end-point, respectively. For simplification, if a topological part consists of a single subpart, the subscript assigned to this subpart is omitted. In addition, brackets of inner matrices are omitted if they have only one element. Originally, the 9$^+$-intersection matrix was introduced to capture such topological DLine-region relations in [6], where 26 relations are identified using the specific constraints on the patterns of the 9$^+$-intersection matrix in Eqn. 3, instead of the universal constraints proposed in this paper.

$$M^{+}(D,R)=\begin{pmatrix} D^{\circ}\cap R^{\circ} & D^{\circ}\cap\partial R & D^{\circ}\cap R^{-} \\ \begin{bmatrix}\partial_1 D\cap R^{\circ}\\ \partial_2 D\cap R^{\circ}\end{bmatrix} & \begin{bmatrix}\partial_1 D\cap \partial R\\ \partial_2 D\cap \partial R\end{bmatrix} & \begin{bmatrix}\partial_1 D\cap R^{-}\\ \partial_2 D\cap R^{-}\end{bmatrix} \\ D^{-}\cap R^{\circ} & D^{-}\cap\partial R & D^{-}\cap R^{-} \end{pmatrix} \tag{3}$$

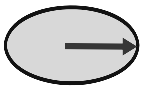
For visualization, the patterns of the 9-/9$^+$-intersection matrix are represented by bitmap-like icons [6, 19]. Each icon is partitioned into nine blocks, which correspond to the nine elements of the 9-intersection matrix or the nine element sets of the 9$^+$-intersection matrix (Figs. 1a-b). In the icon of the 9$^+$-intersection matrix, each block of the icon is further partitioned if the corresponding element set has multiple elements (Fig. 1b). Each block or sub-block is marked out if the corresponding intersection is non-empty. Accordingly, topological relations are distinguished by the icons' marking patterns.



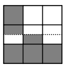
$$\begin{pmatrix} L^{\circ}\cap R^{\circ} & L^{\circ}\cap\partial R & L^{\circ}\cap R^{-} \\ \partial L\cap R^{\circ} & \partial L\cap\partial R & \partial L\cap R^{-} \\ L^{-}\cap R^{\circ} & L^{-}\cap\partial R & L^{-}\cap R^{-} \end{pmatrix} = \begin{pmatrix} \neg\phi & \phi & \phi \\ \neg\phi & \neg\phi & \phi \\ \neg\phi & \neg\phi & \neg\phi \end{pmatrix} \rightarrow$$



(a)



$$\begin{pmatrix} D^{\circ}\cap R^{\circ} & D^{\circ}\cap\partial R & D^{\circ}\cap R^{-} \\ \begin{bmatrix}\partial_1 D\cap R^{\circ}\\ \partial_2 D\cap R^{\circ}\end{bmatrix} & \begin{bmatrix}\partial_1 D\cap \partial R\\ \partial_2 D\cap \partial R\end{bmatrix} & \begin{bmatrix}\partial_1 D\cap R^{-}\\ \partial_2 D\cap R^{-}\end{bmatrix} \\ D^{-}\cap R^{\circ} & D^{-}\cap\partial R & D^{-}\cap R^{-} \end{pmatrix} = \begin{pmatrix} \neg\phi & \phi & \phi \\ \begin{bmatrix}\neg\phi\\ \phi\end{bmatrix} & \begin{bmatrix}\phi\\ \neg\phi\end{bmatrix} & \begin{bmatrix}\phi\\ \phi\end{bmatrix} \\ \neg\phi & \neg\phi & \neg\phi \end{pmatrix} \rightarrow$$



(b)

Fig. 1. Iconic representations of (a) the pattern of the 9-intersection matrix and (b) that of the 9$^+$-intersection matrix

3 Universal Constraints on the 9$^+$-Intersection Matrix

The *primitives* of a spatial object *X* are defined as the subparts of *X*'s interior, boundary, or exterior that are each self-connected and mutually disjoint. For instance, the interior, boundary, and exterior of a DLine in $\mathbf{R}^1$ consist of one, two, and two primitives, respectively, as its boundary consists of two distinctive points (i.e., start-point and end-point) and its exterior consists of two distinctive half-lines (i.e., *front* and *back* exterior subparts). Primitives of spatial objects are classified by their dimension and spatial extent (Table 1). For instance, the primitives that form the interior, boundary, and exterior of the DLine in $\mathbf{R}^1$ are classified into *B-1D* (bounded, non-looped one-dimensional primitive), *0D* (zero-dimensional primitive), and *U-1D* (unbounded one-dimensional primitive), respectively.

By illustrating the class of all primitives that form the interior, boundary, and exterior of a spatial object *X*, as well as the adjacency among these primitives, *X*'s topological structure is represented as a graph. For instance, Fig. 2 illustrates the topological structures of points, DLines, and regions embedded in $\mathbf{R}^1$, $\mathbf{R}^2$, $\mathbf{R}^3$, $\mathbf{S}^1$, and $\mathbf{S}^2$. These topological structures have the following features:

- Every primitive has at least one adjacent primitive; and
- Every pair of adjacent primitives has different dimensions and belongs to different topological parts.

Table 1. Classes of basic primitives

Class	Dim.	Spatial extent	Class	Dim.	Spatial extent
0D	0D	–	B-2D	2D	bounded, non-looped
B-1D	1D	bounded, non-looped	L-2D	2D	spherically looped
L-1D	1D	circularly looped	U-2D	2D	unbounded
U-1D	1D	unbounded	B-3D	3D	bounded non-looped
			U-3D	3D	unbounded

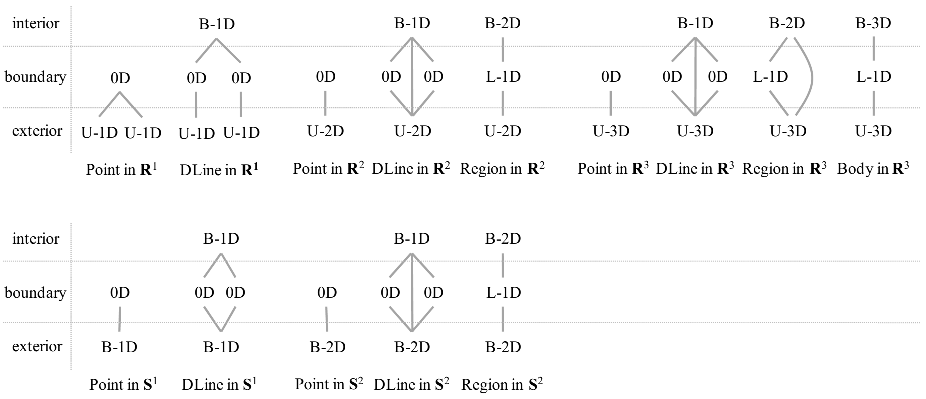
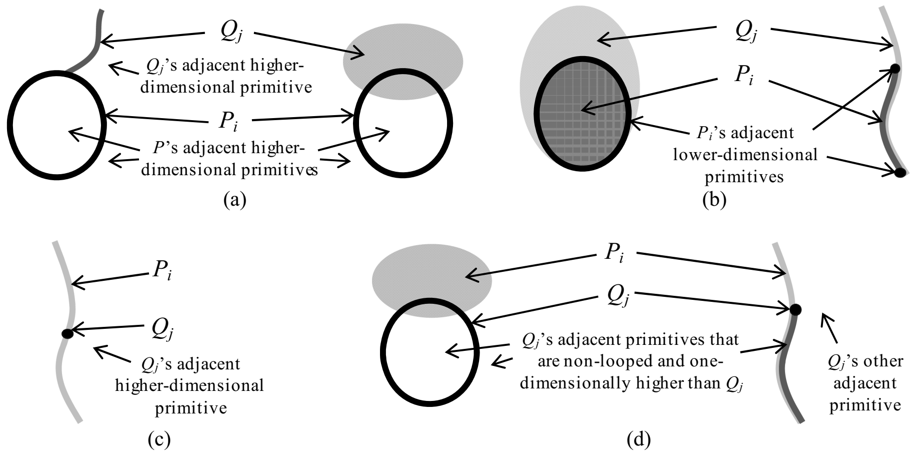


Fig. 2. Topological structures of points, DLines, and regions embedded in $\mathbf{R}^1$, $\mathbf{R}^2$, $\mathbf{R}^3$, $\mathbf{S}^1$, and $\mathbf{S}^2$, based on the class and adjacency of their primitives

Assume that two spatial objects *A* and *B* are embedded in the space *S*. Then, *A*'s primitives and *B*'s primitives must satisfy the following nine conditions (see Appendix for their proofs):

- Condition 1: Each of *A*'s primitives intersects with at least one of *B*'s primitives, and vice versa.
- Condition 2: If *A* has a zero-dimensional primitive (0D), then it intersects with only one of *B*'s primitives, and vice versa.
- Condition 3: If *A*'s primitive P_i intersects with more than one of *B*'s primitives, then these primitives jointly form a connected point set, and vice versa.
- Condition 4: *A*'s bounded primitives, either looped or non-looped, cannot contain a *B*'s unbounded primitives, and vice versa.
- Condition 5: *A*'s non-looped primitives cannot contain a *B*'s looped primitive of the same dimension, and vice versa.

- Condition 6: If A's primitive P_i intersects with B's primitive Q_j, then all of P_i's adjacent higher-dimensional primitives intersect with Q_j or at least one of Q_j's adjacent higher-dimensional primitives, and vice versa (Fig. 3a).
- Condition 7: If A's primitive P_i is contained by B's primitive Q_j and they belong to the same primitive class, then at least one of P_i's adjacent lower-dimensional primitives intersects with Q_j, and vice versa (Fig. 3b).
- Condition 8: If A's primitive P_i intersects with B's primitive Q_j that is lower dimensional than P_i, then P_i intersects with at least one of Q_j's adjacent higher-dimensional primitives (Fig. 3c) or at least two of them including one of Q_j's adjacent primitives that are bounded, non-looped, and one-dimensionally higher than Q_j (Fig. 3d), and vice versa.
- Condition 9: If A has only one unbounded primitive whose dimension is the same with S, then this primitive intersects with all of B's unbounded primitives, and vice versa.



Q_j
Q_j's adjacent higher-dimensional primitive
P_i
P's adjacent higher-dimensional primitives
(a)
Q_j
P_i
P_i's adjacent lower-dimensional primitives
(b)
P_i
Q_j
Q_j's adjacent higher-dimensional primitive
(c)
P_i
Q_j
Q_j's adjacent primitives that are non-looped and one-dimensionally higher than Q_j
Q_j's other adjacent primitive
(d)

Fig. 3. Intersection of two primitives P_i and Q_j determines the presence of intersections of P_i's certain adjacent primitives and Q_j or Q_j's adjacent primitives

When the 9^+-intersection illustrates the intersections of A's primitives and B's primitives, Conditions 1-9 serve as the constraints on the patterns of the 9^+-intersection matrix. For instance, the 9^+-intersection matrix in Eqn. 3 illustrates the intersections of the primitives of a DLine D (D°, $\partial_1 D$, $\partial_2 D$ and D^-) and the primitives of a region R (R°, ∂R and R^-). This matrix may distinguish 4096 patterns, as it has 4×3 two-valued elements, but only 26 patterns among the 4096 patterns satisfy Conditions 1-9. Consequently, we can conclude that these 26 patterns of the 9^+-intersection matrix represent the candidates for the topological relations between a DLine D and a region R.

4 Deriving Topological Relations between Basic Objects

For every possible pair of basic objects (points, DLines, regions, and bodies) embedded in $\mathbf{R}^1$, $\mathbf{R}^2$, $\mathbf{R}^3$, $\mathbf{S}^1$, and $\mathbf{S}^2$, we derived the patterns of the 9^+-intersection matrix that satisfy Conditions 1-9, making use of their structural information illustrated in Fig. 2. Table 2 shows the number of the derived matrix patterns. Since each matrix pattern represents a candidate for topological relations, the number of all possible topological relations between each pair of objects in each space is equal or possibly less than the number in Table 2.

Table 2. Numbers of the derived patterns of the 9^+-intersection matrix for every possible pair of basic objects embedded in $\mathbf{R}^1$, $\mathbf{R}^2$, $\mathbf{R}^3$, $\mathbf{S}^1$, and $\mathbf{S}^2$

	$\mathbf{R}^1$	$\mathbf{R}^2$	$\mathbf{R}^3$	$\mathbf{S}^1$	$\mathbf{S}^2$
Point-Point	6	2	2	2	2
Point-DLine	10	4	4	4	4
Point-Region	–	3	3	–	3
Point-Body	–	–	3	–	–
DLine-DLine	26	80	80	28	80
DLine-Region	–	26	45	–	26
DLine-Body	–	–	26	–	–
Region-Region	–	8	43	–	11
Region-Body	–	–	19	–	–
Body-Body	–	–	8	–	–

For point-point relations in $\mathbf{R}^2$, $\mathbf{R}^3$, $\mathbf{S}^1$, and $\mathbf{S}^2$, the same two patterns of the 9^+-intersection matrix are derived. These two matrix patterns correspond to the two scenarios—two points coincide or not. Meanwhile, point-point relations in $\mathbf{R}^1$ yielded a larger number of matrix patterns, because in $\mathbf{R}^1$ the exterior of each point is subdivided into two primitives (i.e., front and back exterior subparts) and their order influences the distinction of topological relations. Similarly, the same four matrix patterns are derived for point-DLine relations in $\mathbf{R}^2$, $\mathbf{R}^3$, $\mathbf{S}^1$, and $\mathbf{S}^2$, but not for those in $\mathbf{R}^1$. These four matrix patterns correspond to the four scenarios where the point is located at the DLine's interior, exterior, start-point, or end-point. For point-region relations in $\mathbf{R}^2$, $\mathbf{R}^3$, and $\mathbf{S}^2$ and point-body relations in $\mathbf{R}^3$, we derived the same three matrix patterns, which correspond to the three scenarios where the point is located at the interior, exterior, or boundary of the region/body.

For DLine-DLine relations in $\mathbf{R}^1$, we derived 26 patterns of the 9^+-intersection matrix. These 26 patterns correspond to the 26 relations between two directed intervals in $\mathbf{R}^1$ [20], because directed intervals are essentially DLines.

For DLine-DLine relations in $\mathbf{R}^2$, 80 patterns of the 9^+-intersection matrix are derived. Kurata and Egenhofer [5] identified only 68 DLine-DLine relations in $\mathbf{R}^2$, as they used an extension of the 4-intersection. They also speculated that additional 12 relations appear if they distinguish the collapsed configurations (i.e., the configurations where one DLine contains another DLine). Our 80 patterns of the 9^+-intersection matrix successfully distinguish these 68 + 12 = 80 DLine-DLine relations.

For DLine-region relations in $\mathbf{R}^2$, we derived 26 patterns of the 9^+-intersection matrix. This successfully corresponds to the result in [6], which identified 26 DLine-region relations based on the 9^+-intersection.

DLine-DLine relations in $\mathbf{R}^2$, $\mathbf{R}^3$, and $\mathbf{S}^2$ yielded the same 80 matrix patterns. This result indicates that the set of 80 DLine-DLine relations in $\mathbf{R}^2$ are also seen in $\mathbf{R}^3$ and $\mathbf{S}^2$. Similarly, from the result that DLine-region relations in $\mathbf{R}^2$, DLine-region relations in $\mathbf{S}^2$, and DLine-body relations in $\mathbf{R}^3$ has yielded the same 26 matrix patterns, it is concluded that the set of 26 DLine-region relations in $\mathbf{R}^2$ are also seen in $\mathbf{S}^2$, and have a one-to-one correspondence with DLine-body relations in $\mathbf{R}^3$. This correspondence stems from the structural similarity between a region in $\mathbf{R}^2$ and a body in $\mathbf{R}^3$ (Fig. 2).

For DLine-Region relations in $\mathbf{R}^3$, we derived 45 patterns of the 9^+-intersection matrix, among which 26 patterns are identical to the 26 matrix patterns derived for DLine-Region relations in $\mathbf{R}^2$. We confirmed that each of the remaining 19 patterns has at least one geometrical realization in $\mathbf{R}^3$ (Fig. 4). Consequently, we identified that topological DLine-Region relations in $\mathbf{R}^3$ consist of 19 relations peculiar to $\mathbf{R}^3$ (Fig. 4) and 26 relations common to $\mathbf{R}^2$, $\mathbf{R}^3$, and $\mathbf{S}^2$ (see [6] for the list). Such DLine-Region relations in $\mathbf{R}^3$ are useful for categorizing the movement patterns in association with region-like landmarks—for instance, how a bird moves around a pond. The newly identified 19 relations capture the bird's movement patterns that realize only in a three-dimensional space.

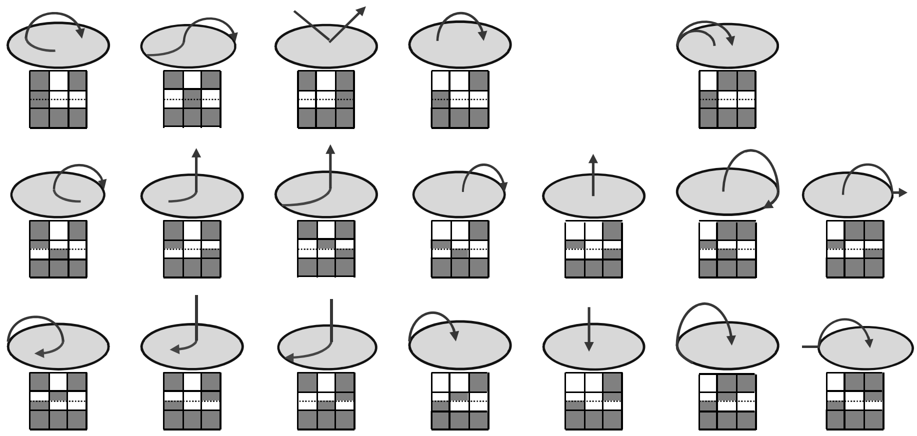


Fig. 4. 19 topological DLine-region relations, which are peculiar to $\mathbf{R}^3$

For region-region relations in $\mathbf{R}^2$, eight patterns of the 9^+-intersection matrix are derived. In this case, the matrix is essentially the 9-intersection matrix, because no topological part of a region is subdivided into multiple primitives. These eight matrix patterns are exactly the same as those derived in [2]. Similarly, the following correspondences to the findings of the previous studies are observed:

- The eleven matrix patterns derived for region-region relations in **S**^{2} are the same as those derived in [11];
- The nineteen matrix patterns derived for region-body in **R**^{3} are the same as those derived in [9]; and
- The eight matrix patterns derived for body-body relations in **R**^{3} are the same as those derived in [9].

Meanwhile, for region-region relations in **R**^{3}, we derived 43 patterns of the 9-intersection matrix, while only 38 patterns are reported in [9]. Through the comparison the matrix patterns, we identified five missing relations (Fig. 5). In [21], 43 region-region relations in **R**^{3} are derived based on another spatial model, called *Dimensional Model* [18]. The 43 region-region relations in the Dimensional Model are equivalent to the region-region relations characterized by our 43 patterns of the 9-intersection matrix.

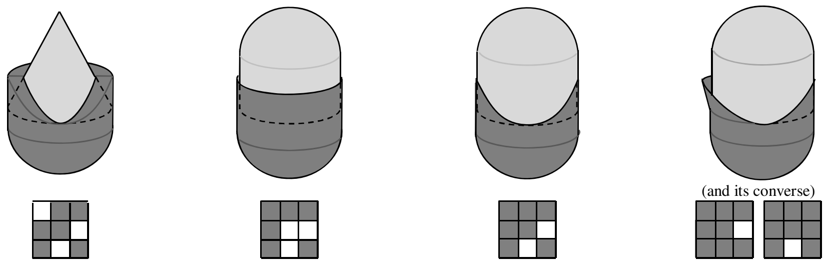


Fig. 5. Five topological region-region relations in **R**^{3} that should be added to the list in [9]

5 Topological Relations Derived by Matrix Conversion

A unique feature of the 9$^+$-intersection is that the 9$^+$-intersection matrix can be converted to the 9-intersection matrix, simply by integrating the elements of its nine element sets by union operation (Fig. 6). If the original 9$^+$-intersection matrix captures the topological relation between a DLine and another object *X*, the 9-intersection matrix derived by the conversion captures the topological line-*X* relations, as the distinction of the DLine's start-point and end-point is lost. Similarly, for the relations in **R**^{1}, the distinction of objects' front and back exterior subparts is lost by the matrix conversion (Fig. 6b). Making use of this matrix conversion, the sets of candidates for topological line-*X* relations and the modified candidate sets for topological relations in **R**^{1} are derived from the matrix patterns derived in Section 4. Table 3 shows the number of the derived patterns of the 9-intersection matrix.

For line-line relations in **R**^{1}, eight patterns of the 9-intersection matrix are derived. These eight patterns correspond to the eight line-line relations in **R**^{1}, identified in [22]. In addition, these eight matrix patterns are the same as the matrix patterns for region-region relations in **R**^{2}. This indicates a one-to-one correspondence between the eight line-line relations in **R**^{1} and the eight region-region relations in **R**^{2}. Similarly, we found a one-to-one correspondence between the eleven line-line relations in **S**^{1} and the eleven region-region relations in **S**^{2}.

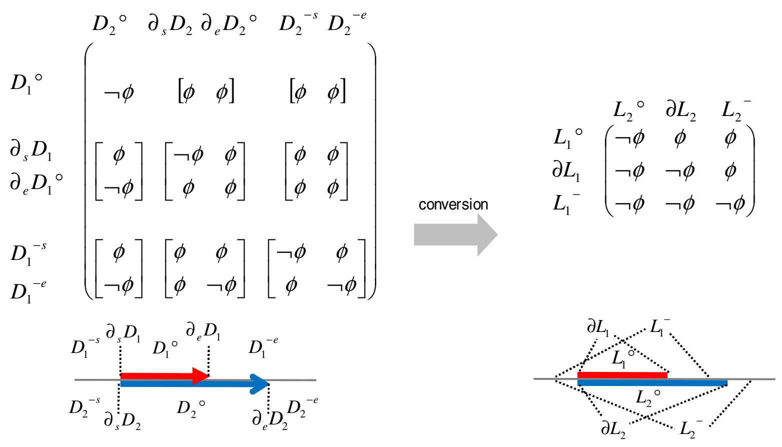


Fig. 6. Conversion from the 9⁺-intersection matrix to the 9-intersection matrix. In the left figure, DLines D_1 and D_2 are represented by two arrows.

Table 3. Numbers of the patterns of the 9-intersection matrix derived by matrix conversion

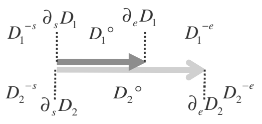
	$\mathbf{R}^1$	$\mathbf{R}^2$	$\mathbf{R}^3$	$\mathbf{S}^1$	$\mathbf{S}^2$
Point-Point	2	–	–	–	–
Point-Line	3	–	–	–	–
Line-Line	8	33	33	11	33
Line-Region	–	19	31	–	19
Line-Body	–	–	19	–	–

For line-line and line-region relations in $\mathbf{R}^2$, we derived 33 and 19 patterns of the 9-intersection matrix. These two sets of matrix patterns are the same as those in [2]. Similarly, for line-line, line-region, and line-body relations in $\mathbf{R}^3$, we derived 33, 31, and 19 patterns of the 9-intersection matrix, which are the same as those in [9].

The 9^+-intersection matrix allows another type of matrix conversion when both objects have topological parts that can be subdivided into multiple primitives. This conversion removes the subdivision of topological parts with respect to only one of the two objects (Fig. 7). Accordingly, the matrix derived by the conversion is still the 9^+-intersection matrix. Making use of this matrix conversion, we derived additional sets of patterns of the 9^+-intersection matrix from those of the 9^+-intersection matrix in Section 4. Table 4 shows the numbers of the derived matrix patterns.

For point-point relations in $\mathbf{R}^1$, three patterns of the 9^+-intersection matrix are derived. These three matrix patterns correspond to the three scenarios where one point precedes, coincides, or succeeds another point on the same axis. In the previous step, we derived only two patterns for the same relations, because the 9-intersection matrix does not distinguish whether one point precedes or succeeds another point. For the same reason, the number of the patterns of the 9^+-intersection matrix derived for point-line relation in $\mathbf{R}^1$ is larger than that of the 9-intersection matrix.

$$\begin{array}{c|ccc} & D_2{}^\circ & \partial_s D_2 \ \partial_e D_2{}^\circ & D_2{}^{-s} \ D_2{}^{-e} \\ \hline D_1{}^\circ & \neg\phi & \begin{bmatrix}\phi & \phi\end{bmatrix} & \begin{bmatrix}\phi & \phi\end{bmatrix} \\ \begin{matrix}\partial_s D_1 \\ \partial_e D_1{}^\circ\end{matrix} & \begin{bmatrix}\phi \\ \neg\phi\end{bmatrix} & \begin{bmatrix}\neg\phi & \phi \\ \phi & \phi\end{bmatrix} & \begin{bmatrix}\phi & \phi \\ \phi & \phi\end{bmatrix} \\ \begin{matrix}D_1{}^{-s} \\ D_1{}^{-e}\end{matrix} & \begin{bmatrix}\phi \\ \neg\phi\end{bmatrix} & \begin{bmatrix}\phi & \phi \\ \phi & \neg\phi\end{bmatrix} & \begin{bmatrix}\neg\phi & \phi \\ \phi & \neg\phi\end{bmatrix} \end{array} \rightarrow \begin{array}{c|ccc} & L_2{}^\circ & \partial L_2{}^\circ & L_2{}^- \\ \hline D_1{}^\circ & \neg\phi & \phi & \phi \\ \begin{matrix}\partial_s D_1 \\ \partial_e D_1{}^\circ\end{matrix} & \begin{bmatrix}\phi \\ \neg\phi\end{bmatrix} & \begin{bmatrix}\neg\phi \\ \phi\end{bmatrix} & \begin{bmatrix}\phi \\ \phi\end{bmatrix} \\ \begin{matrix}D_1{}^{-s} \\ D_1{}^{-e}\end{matrix} & \begin{bmatrix}\phi \\ \neg\phi\end{bmatrix} & \begin{bmatrix}\phi \\ \neg\phi\end{bmatrix} & \begin{bmatrix}\neg\phi \\ \neg\phi\end{bmatrix} \end{array}$$



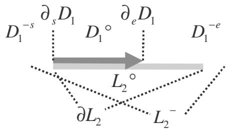


Fig. 7. Conversion of the 9+-intersection matrix that removes the subdivision of the second object's topological parts

Table 4. Numbers of the patterns of the 9+-intersection matrix that are newly derived by the matrix conversion in Fig. 7

	$\mathbf{R}^1$	$\mathbf{R}^2$	$\mathbf{R}^3$	$\mathbf{S}^1$	$\mathbf{S}^2$
Point-Point	3	–	–	–	–
Point-Line	5	–	–	–	–
DLine-Line	13	49	49	16	49

For DLine-line relations in $\mathbf{R}^1$, 13 patterns of the 9+-intersection matrix are derived. Interestingly, these 13 matrix patterns correspond to the 13 interval relations in Allen's interval algebra [8], because DLine-Line relations in $\mathbf{R}^1$ essentially illustrate how one line (interval) extends with respect to another line (interval) on the same axis with the distinction of front and back. Similarly, for DLine-Line relations in $\mathbf{S}^1$, we derived 16 matrix patterns, which correspond to the 16 interval relations in a cyclic temporal frame identified in [10].

For DLine-line relations in $\mathbf{R}^2$, $\mathbf{R}^3$, and $\mathbf{S}^2$, the same 49 patterns of the 9+-intersection matrix are derived. We confirmed that each of these 49 matrix patterns has at least one geometric realization (Fig. 8). Thus, it is concluded that these 49 matrix patterns represent the set of all possible DLine-line relations in $\mathbf{R}^2$, $\mathbf{R}^3$, and $\mathbf{S}^2$ under the 9+-intersection. Such DLine-line relations are useful for categorizing the movement patterns associated with a linear landmark, such as a wall or a trench.

Finally, through the comparison of Tables 2-4, we found the following features:

- The number of matrix patterns for line-line/DLine-line/DLine-DLine relations in $\mathbf{S}^1$ are larger than that for the counterparts in $\mathbf{R}^1$, because $\mathbf{S}^1$ allows the relation where one line/DLine includes the entire exterior of another line/DLine, but $\mathbf{R}^1$ does not.

- The number of topological relations between two spatial objects is invariant to the embedding space, as long as these two objects are lower dimensional than the space. One exception is point-point relations in $\mathbf{R}^1$ distinguished by the 9^+-intersection (Table 2), because the distinction of the points' front and back exteriors increases the number of point-point relations.

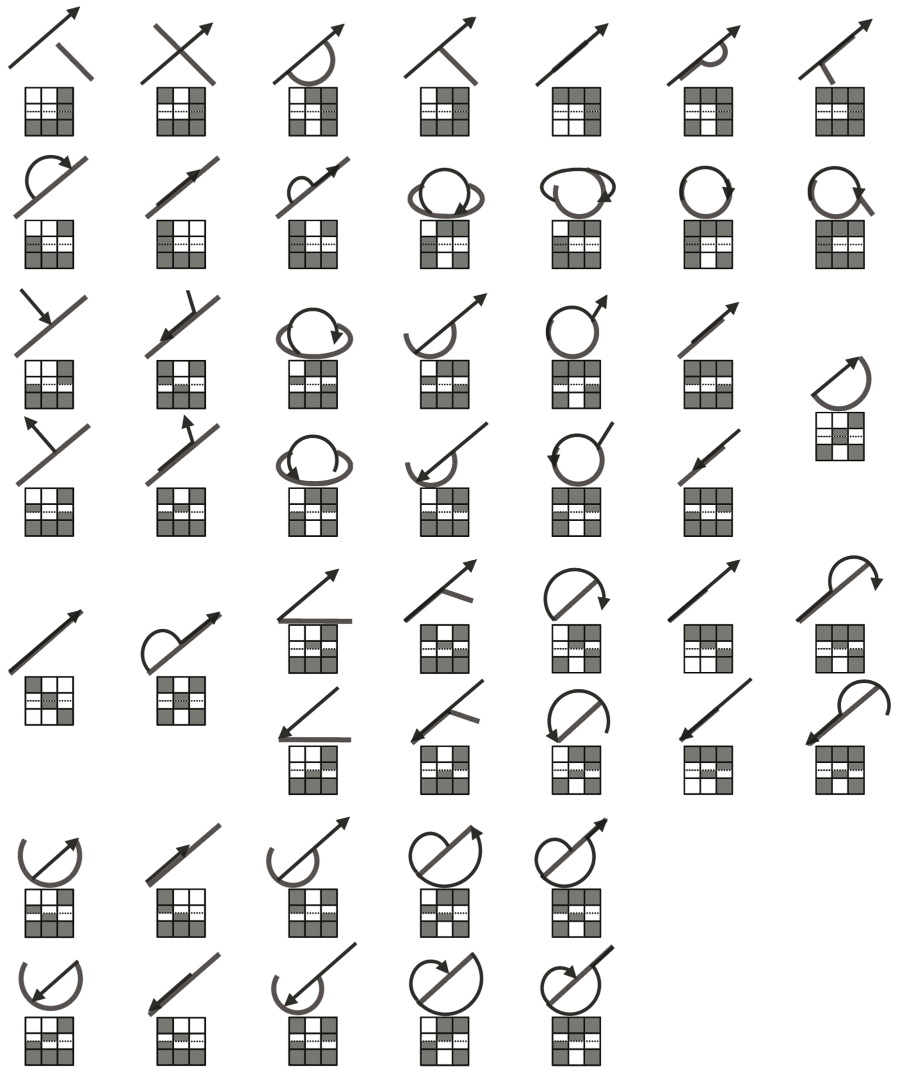


Fig. 8. 49 topological DLine-line relations, common to $\mathbf{R}^2$, $\mathbf{R}^3$, and $\mathbf{S}^2$, each with an example of geometric realizations

6 Deriving Topological Relations Related to Complicated Objects

The combination of the 9^+-intersection and the universal constraints are effective also for deriving the possible topological relations between a basic object and a more complicated object, or even the relations between two complicated objects (here, complicated objects mean the objects that are derived by a set operation on multiple basic objects). As a demonstration, this section derives all possible topological relations between a simple region (so far we have called this a region) and a simple region with a hole. For simplification, these relations are called *topological region-region$^{+1\mathrm{H}}$ relations*. In [7], 23 topological region-region$^{+1\mathrm{H}}$ relations in $\mathbf{R}^2$ are already identified, but those in $\mathbf{S}^2$ are not yet.

A simple region with a hole X is defined as the difference between two simple regions X^* and X_H, where X^* contains X_H entirely. X^* is called X's *generalized region*, while X_H is called X's *hole* [23]. A simple region with a hole in $\mathbf{S}^2$ may represent a ring-like object on a sphere (Fig. 9a) or a belt-like object that surrounds the sphere (Fig. 9b) Accordingly, region-region$^{+1\mathrm{H}}$ relations in $\mathbf{S}^2$ can be used for modeling, for instance, the spatial arrangement of an island and a typhoon, that of the iris and the covered surface of a human eye, or that of the Earth's surface receiving sunlight and a certain latitude zone.

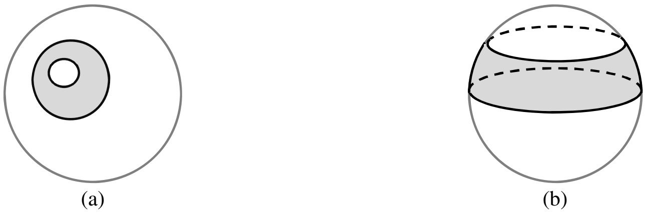


Fig. 9. Examples of simple regions with a hole embedded in $\mathbf{S}^2$

The topological relation between a simple region A and a simple region with a hole B are characterized by the 9^+-intersection matrix in Eqn. 4, where $\partial_1 B$, $\partial_2 B$, B^{-1}, and B^{-2} are B's outer boundary, hole-side boundary, outer exterior, and hole-side exterior, respectively. In $\mathbf{S}^2$, the outer exterior and the hole-side exterior have no geometric difference (Fig. 9b) and, accordingly, their distinction depends on the observer's view.

$$M^+(A,B)=\begin{pmatrix} A^\circ \cap B^\circ & [A^\circ \cap \partial_1 B & A^\circ \cap \partial_2 B] & [A^\circ \cap B^{-1} & A^\circ \cap B^{-2}] \\ \partial A \cap B^\circ & [\partial A \cap \partial_1 B & \partial A \cap \partial_2 B] & [\partial A \cap B^{-1} & \partial A \cap B^{-2}] \\ A^- \cap B^\circ & [A^- \cap \partial_1 B & A^- \cap \partial_2 B] & [A^- \cap B^{-1} & A^- \cap B^{-2}] \end{pmatrix} \quad (4)$$

Instead of the 9+-intersection matrix in Eqn. 4, the same topological region-region$^{+1\mathrm{H}}$ relation is characterized by a pair of the 9-intersection matrices, which represent the topological region-region relation between A and B's generalized region B^* and that between A and B's hole B_H, respectively [7, 23]. The patterns of these two 9-intersection matrices are determined uniquely from the pattern of the corresponding 9^+-intersection matrix (Fig. 10).

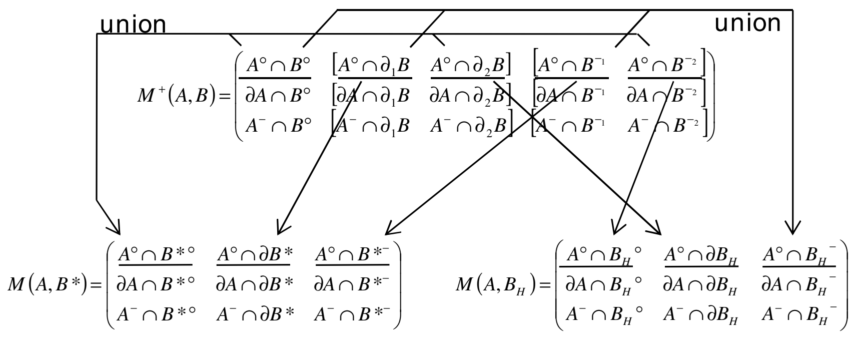

union
union
$M^{+}(A,B)=\begin{pmatrix} A^\circ\cap B^\circ & [A^\circ\cap\partial_1 B & A^\circ\cap\partial_2 B] & [A^\circ\cap B^{-1} & A^\circ\cap B^{-2}] \\ \partial A\cap B^\circ & [\partial A\cap\partial_1 B & \partial A\cap\partial_2 B] & [\partial A\cap B^{-1} & \partial A\cap B^{-2}] \\ A^-\cap B^\circ & [A^-\cap\partial_1 B & A^-\cap\partial_2 B] & [A^-\cap B^{-1} & A^-\cap B^{-2}] \end{pmatrix}$
$M(A,B^*)=\begin{pmatrix} A^\circ\cap B^{*\circ} & A^\circ\cap\partial B^* & A^\circ\cap B^{*-} \\ \partial A\cap B^{*\circ} & \partial A\cap\partial B^* & \partial A\cap B^{*-} \\ A^-\cap B^{*\circ} & A^-\cap\partial B^* & A^-\cap B^{*-} \end{pmatrix}$
$M(A,B_H)=\begin{pmatrix} A^\circ\cap B_H^\circ & A^\circ\cap\partial B_H & A^\circ\cap B_H^- \\ \partial A\cap B_H^\circ & \partial A\cap\partial B_H & \partial A\cap B_H^- \\ A^-\cap B_H^\circ & A^-\cap\partial B_H & A^-\cap B_H^- \end{pmatrix}$

Fig. 10. Conversion from the 9^+-intersection matrix for a topological region-region^{+1H} relation to a pair of the 9-intersection matrices. This figure shows the conversion of the elements in the matrix's first row. The elements in the second and third rows are converted in the same way.

Fig. 11 shows the topological structures of simple regions and simple regions with a hole embedded in $\mathbf{R}^2$ and $\mathbf{S}^2$. Based on this structural information and Conditions 1-9 (i.e., universal constraints), we computationally derived 23 and 37 matrix patterns for topological region-region^{+1H} relations in $\mathbf{R}^2$ and $\mathbf{S}^2$, respectively. It is confirmed by the matrix conversion in Fig. 10 that the former 23 matrix patterns perfectly correspond to the 23 region-region^{+1H} relations in $\mathbf{R}^2$ identified in [7]. Among the latter 37 matrix patterns, 23 patterns are the same as those for region-region^{+1H} relations in $\mathbf{R}^2$. Meanwhile, we confirmed that each of the remaining 14 matrix patterns has at least one geometric realization in $\mathbf{S}^2$ (Fig. 12). Thus, it is confirmed that topological region-region^{+1H} relations in $\mathbf{S}^2$ consist of 14 relations peculiar to $\mathbf{S}^2$ (Fig. 12) and 23 relations that are also seen in $\mathbf{R}^2$ (see [7] for the list).

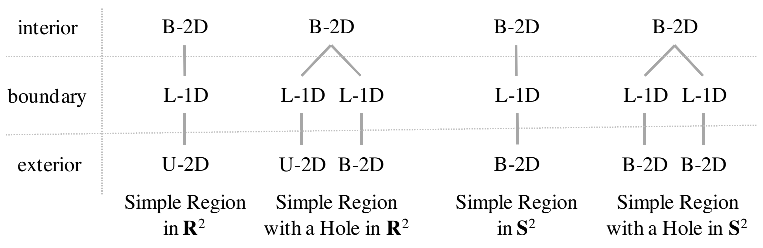

interior
boundary
exterior
B-2D
L-1D
U-2D
Simple Region in $\mathbf{R}^2$
B-2D
L-1D L-1D
U-2D B-2D
Simple Region with a Hole in $\mathbf{R}^2$
B-2D
L-1D
B-2D
Simple Region in $\mathbf{S}^2$
B-2D
L-1D L-1D
B-2D B-2D
Simple Region with a Hole in $\mathbf{S}^2$

Fig. 11. Topological structures of simple regions and simple regions with a hole, embedded in $\mathbf{R}^2$ and $\mathbf{S}^2$

In the same way, it is possible to derive the candidates for topological relations between a basic object and a complicated object (e.g., DLine-region^{+1H} relations), as well as the relations between two complicated objects (e.g., region^{+1H}-region^{+1H} relations).

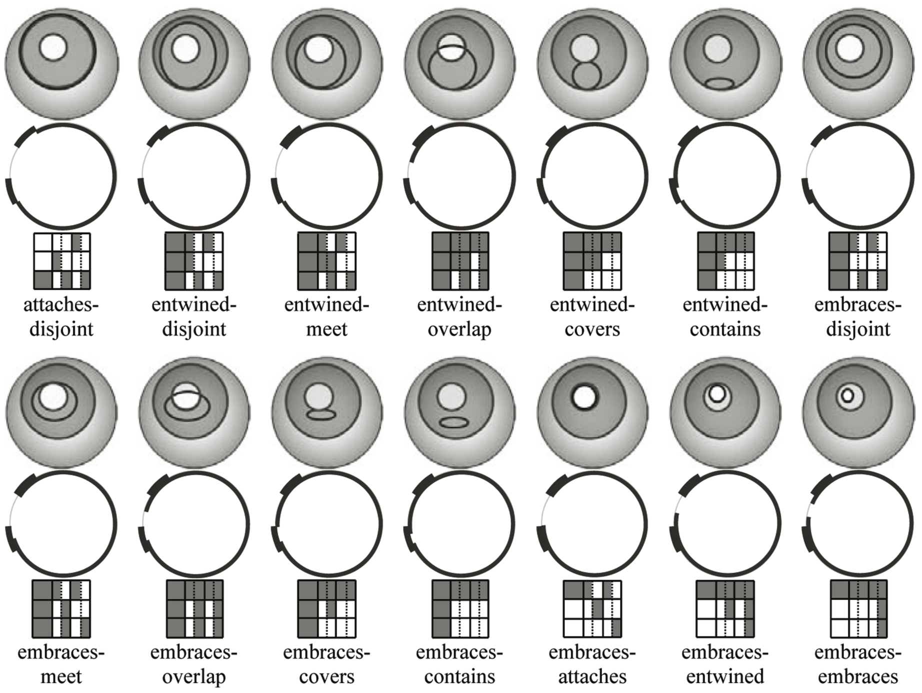


Fig. 12. 14 topological relations between a region *A* and a region with a hole *B*, which are peculiar to $\mathbf{S}^2$, together with an example configuration and its cross section. The name given to each relation combines the name of the topological relation between *A* and *B*'s generalized region and that between *A* and *B*'s hole, following the notation of [7].

7 Conclusions

Topological relations, which concern how two objects intersect with each other, have been studied for decades in pursuit of cognitively adequate models of the objects' spatial arrangement. The 9^+-intersection captures such topological relations in a systematic way using a nested matrix, called the 9^+-intersection matrix. This paper developed a set of universal constraints on the patterns of the 9^+-intersection matrix. These constraints allow us to derive the candidates for the topological relations between a given pair of objects, regardless of their geometric types. The combination of the 9^+-intersection and the universal constraints is, therefore, highly useful when we study new sets of topological relations. Indeed, we newly identified DLine-Region relations in $\mathbf{R}^3$, DLine-Line relations in $\mathbf{R}^2$, $\mathbf{R}^3$, and $\mathbf{S}^2$, and region-region^{+1H} relations in $\mathbf{S}^2$, as well as found that many sets of topological relations in $\mathbf{S}^1$ and $\mathbf{S}^2$ are equivalent to those in $\mathbf{R}^1$ and $\mathbf{R}^2$.

A remaining issue is to answer whether our universal constraints are always sufficient, in the sense that every pattern of the 9^+-intersection matrix, determined by the universal constraints, always has at least one geometric realization. The constraints' sufficiency is empirically confirmed with respect to the relation between every possible pair of basic objects embedded in $\mathbf{R}^1$, $\mathbf{R}^2$, $\mathbf{R}^3$, $\mathbf{S}^1$, and $\mathbf{S}^2$. For the relations between complicated objects, however, it becomes hard to check the validity of all derived matrix patterns due to its large number. Thus, a mathematical examination of the constraints' sufficiency is left for future work.

Acknowledgements

Yohei Kurata's work is supported by DFG (Deutsche Forschungsgemeinschaft) through the Collaborative Research Center SFB/TR 8 Spatial Cognition. The author appreciates Dr. Max J. Egenhofer, Dr. Roland Billen, and Dr. Tetsushi Ito for their helpful advice and information.

References

1. Egenhofer, M., Franzosa, R.: Point-Set Topological Spatial Relations. International Journal of Geographical Information Systems 5, 161–174 (1991)
2. Egenhofer, M., Herring, J.: Categorizing Binary Topological Relationships between Regions, Lines and Points in Geographic Databases. In: Egenhofer, M., Herring, J., Smith, T., Park, K. (eds.): NCGIA Technical Reports 91-7. National Center for Geographic Information and Analysis, Santa Barbara, CA, USA (1991)
3. Randell, D., Cui, Z., Cohn, A.: A Spatial Logic Based on Regions and Connection. In: Nebel, B., Rich, C., Swarout, W. (eds.) 3rd International Conference on Knowledge Representation and Reasoning, pp. 165–176. Morgan Kaufmann, San Francisco (1992)
4. Schneider, M., Behr, T.: Topological Relationships between Complex Spatial Objects. ACM Transactions on Database Systems 31, 39–81 (2006)
5. Kurata, Y., Egenhofer, M.: The Head-Body-Tail Intersection for Spatial Relations between Directed Line Segments. In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 269–286. Springer, Heidelberg (2006)
6. Kurata, Y., Egenhofer, M.: The 9^+-Intersection for Topological Relations between a Directed Line Segment and a Region. In: Gottfried, B. (ed.) 1st International Symposium for Behavioral Monitoring and Interpretation, pp. 62–76 (2007)
7. Egenhofer, M., Vasardani, M.: Spatial Reasoning with a Hole. In: Winter, S., Duckham, M., Kulik, L., Kuipers, B. (eds.) COSIT 2007. LNCS, vol. 4736, pp. 303–320. Springer, Heidelberg (2007)
8. Allen, J.: An Interval-Based Representation of Temporal Knowledge. In: Hayes, P. (ed.): 7th International Joint Conference on Artificial Intelligence, pp. 221-226 (1981)
9. Zlatanova, S.: On 3D Topological Relationships. In: 11th International Workshop on Database and Expert Systems Applications, pp. 913–924. IEEE Computer Society, Los Alamitos (2000)
10. Hornsby, K., Egenhofer, M., Hayes, P.: Modeling Cyclic Change. In: Chen, P., Embley, D., Kouloumdjian, J., Liddle, S., Roddick, J. (eds.) TABLEAUX 1997. LNCS, vol. 1227, pp. 98–109. Springer, Heidelberg (1999)
11. Egenhofer, M.: Spherical Topological Relations. Journal on Data Semantics III, 25–49 (2005)
12. Egenhofer, M., Sharma, J.: Topological Relations between Regions in R^2 and Z^2. In: Abel, D.J., Ooi, B.-C. (eds.) SSD 1993. LNCS, vol. 692, pp. 316–336. Springer, Heidelberg (1993)

13. Mark, D.: Calibrating the Meanings of Spatial Predicates from Natural Language: Line-Region Relations. In: Waugh, T., Healey, R. (eds.) 6th International Symposium on Spatial Data Handling, pp. 538–553. Taylor Francis (1994)
14. Nedas, K., Egenhofer, M., Wilmsen, D.: Metric Details of Topological Line-Line Relations. International Journal of Geographical Information Science 21, 21–48 (2007)
15. Clementini, E., Di Felice, P.: A Model for Representing Topological Relationships between Complex Geometric Features in Spatial Databases. Information Science 90, 121–136 (1996)
16. Alexandroff, P.: Elementary Concepts of Topology. Dover Publications, Mineola (1961)
17. Egenhofer, M., Franzosa, R.: On the Equivalence of Topological Relations. International Journal of Geographical Information Systems 9, 133–152 (1995)
18. Billen, R., Zlatanova, S., Mathonet, P., Boniver, F.: The Dimensional Model: A Framework to Distinguish Spatial Relationships. In: Richardson, D., van Oosterom, P. (eds.) 10th International Symposium on Spatial Data Handling, pp. 285–298. Springer, Heidelberg (2002)
19. Mark, D., Egenhofer, M.: Modeling Spatial Relations between Lines and Regions: Combining Formal Mathematical Models and Human Subjects Testing. Cartography and Geographical Information Systems 21, 195–212 (1994)
20. Renz, J.: A Spatial Odyssey of the Interval Algebra: 1. Directed Intervals. In: Nebel, B. (ed.) 7th International Joint Conference on Artificial Intelligence, pp. 51–56. Morgan Kaufmann, San Francisco (2001)
21. Billen, R.: Nouvelle Perception De La Spatialité Des Objets Et De Leurs Relations. Dévelopment D'une Modélisation Tridimensionnelle De L'information Spatiale. Department of Geography, Ph.D. Thesis. University of Liège, Liège, Belgium (2002)
22. Pullar, D., Egenhofer, M.: Towards Formal Definitions of Topological Relations among Spatial Objects. In: Marble, D. (ed.) 3rd International Symposium on Spatial Data Handling, pp. 225–241 (1988)
23. Egenhofer, M., Clementini, E., Di Felice, P.: Topological Relations between Regions with Holes. International Journal of Geographical Information Science 8, 129–142 (1994)

Appendix: Proofs of Conditions 1-9 in Section 3

Conditions 1-3

These three conditions are satisfied because the primitives of a spatial object are jointly exhaustive, mutually exclusive, and self-connected, respectively.

Conditions 4-5

These two conditions are basic properties of topology.

Condition 6

Let P_k be a P_i's adjacent higher-dimensional primitive. P_k is adjacent to $P_i \cap Q_j$ as well. On the other hand, $P_i \cap Q_j$ is surrounded entirely by the set of all Q_j's adjacent higher-dimensional primitives and $Q_j \setminus P_i$. Accordingly, P_k intersects with at least one of Q_j's adjacent higher-dimensional primitives or Q_j.

Condition 7

Let us consider a subspace of S, called S', which embeds Q_j and whose dimension is the same with Q_j. Since Q_j contains P_i, S' also embeds P_i. P_i's adjacent primitives in S', which are always lower-dimensional than P_i, is identical with the set of P_i's adjacent lower-dimensional primitives in S. Since Q_j contains P_i in S', Q_j intersect with at least one of P_i's adjacent primitives in S'. Accordingly, Q_j intersect with at least one of P_i's adjacent lower-dimensional primitives in S.

Condition 8

Let $\boldsymbol{Q}$ be the set of all of Q_j's adjacent higher-dimensional primitives. $P_i \cap Q_j$ is surrounded entirely by $\boldsymbol{Q}$ and $Q_j \setminus P_i$. Accordingly, P_i, which contains $P_i \cap Q_j$ and higher dimensional than $Q_j \setminus P_i$, intersects with at least one of primitives in $\boldsymbol{Q}$. Let us assume that Q_j has an adjacent primitive that is bounded, non-looped, and one-dimensionally higher than Q_j, called Q_k. Q_k is an element of $\boldsymbol{Q}$. Let S' be a sub-space of S, which embeds P_i and whose dimension is the same with P_i. Since Q_j cannot split the Q_k, Q_j must be located at the end of Q_k. Accordingly, $P_i \cap Q_j$ is not surrounded entirely by Q_k in S', but by the combination of Q_k, some other elements in $\boldsymbol{Q}$, and $Q_j \setminus P_i$ in S'. On the other hand, $P_i \cap Q_j$ is surrounded entirely by $P_i \setminus Q_j$ and $Q_j \setminus P_i$ in S'. Consequently, if Q_k exists, P_i intersects with not only Q_k, but also another element in $\boldsymbol{Q}$.

Condition 9

Let P_i be A's only one unbounded primitive whose dimension is the same with S. Since P_i's complement $\overline{P_i}$ is bounded, each of B's unbounded primitive is not entirely included in $\overline{P_i}$. Consequently, P_i intersects with B's unbounded primitives. Note P_i refers to the exterior of a spatial object in $\mathbf{R}^n$ ($n \geq 2$).

Decentralized Movement Pattern Detection amongst Mobile Geosensor Nodes

Patrick Laube[1], Matt Duckham[1], and Thomas Wolle[2]

[1] Department of Geomatics, The University of Melbourne, VIC 3010, Australia
plaube@unimelb.edu.au
[2] NICTA Sydney, Locked Bag 9013, Alexandria, NSW 1435, Australia

Abstract. Movement patterns, like *flocking* and *converging*, *leading* and *following*, are examples of high-level process knowledge derived from low-level trajectory data. Conventional techniques for the detection of movement patterns rely on centralized "omniscient" computing systems that have global access to the trajectories of mobile entities. However, in decentralized spatial information processing systems, exemplified by wireless sensor networks, individual processing units may only have access to *local* information about other individuals in their immediate spatial vicinity. Where the individuals in such decentralized systems are mobile, there is a need to be able to detect movement patterns using collaboration between individuals, each of which possess only partial knowledge of the global system state. This paper presents an algorithm for decentralized detection of the movement pattern *flock*, with applications to mobile wireless sensor networks. The algorithm's reliability is evaluated through testing on simulated trajectories emerging from unconstrained random movement and correlated random walk.

1 Introduction

Movement patterns represent high-level process knowledge derived from low-level trajectory data [1]. They are the spatiotemporal "trace" left behind by the behavior of moving entities [2]. Examples of movement patterns include *flocking* as in a "mob" of sheep [3], *leading* and *following* found in group dynamics [4, 5], or *converging* and *diverging* of pedestrians in crowding scenarios [6, 7]. Figure 1 illustrates the movement pattern of a prototypical "flock". The figure shows that at a particular time instant t, four moving entities a, b, d, e are in close spatial proximity (all lie within a circle of radius p). A commonsense interpretation of a flock is where mobile individuals maintain such spatial proximity over an extended period of time.

Wireless sensor networks (WSN) are increasingly applied in a dynamic context. So-called *mobile wireless sensor networks* (mWSN) provide new opportunities for monitoring and understanding coordinated movement processes, with a wide range of applications including "smart" farming [8], emergency response [9], robotics [10], and in-car navigation [11]. Conventional techniques for the detection of movement patterns rely on centralized "omniscient" computing databases

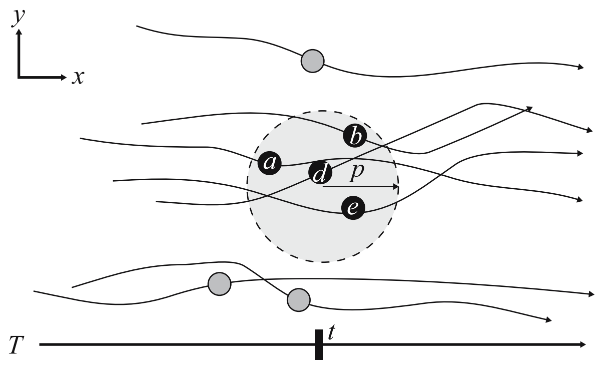


Fig. 1. A "flock" can be defined as a set of entities moving in spatial proximity for some specified time period

and systems that have global access to the trajectories of mobile entities. However, in decentralized spatial information processing systems, like mWSNs, individual processing units may only have access to *local* information about the movement of other individuals in their immediate spatial vicinity. This paper presents a decentralized algorithm that enables moving sensor nodes in an mWSN to infer if they are part of an ongoing flocking movement pattern. The algorithm uses only local collaboration with no central control, and relies on qualitative spatial information (i.e., it does not require precise coordinate information about the location of individuals). Like most tractable centralized flock detection algorithms, our decentralized algorithm generates approximate solutions, so the paper includes an empirical analysis of reliability of the algorithm.

The remainder of this paper is structured as follows. Section 2 reviews the relevant literature on mining movement patterns and mobile wireless sensor networks. Section 3 introduces the underlying model of mobility and flocking in an mWSN. Section 4 presents the algorithm for detecting flocks in an mWSN, termed FLAGS. A series of simulation experiments are used to evaluate the reliability and efficiency of FLAGS in Sect. 5. The paper discusses the findings of this work in Sect. 6 and concludes with final remarks and an outlook in Sect. 7.

2 Background

2.1 Mining Movement Patterns

Various data mining techniques have been used to detect generic movement patterns [2]. In 2004, Laube et al. [12] defined a collection of spatiotemporal patterns based on direction of movement and location, e.g. *flock*, *leadership*, *convergence* and *encounter*, and they gave algorithms to compute them efficiently. Several subsequent articles improved the formalization and the algorithmic discovery of such patterns.

Benkert et al. [13] modified the original definition of a flock to be a set of entities moving close together during a time interval. The applied data mining approach bases on projection of 2D trajectories into multidimensional space and query operations on quadtrees. This approach results in efficient approximation algorithms for finding such flock patterns of a fixed length and a fixed number of flock entities, where the radius is approximated within a factor of 2. For the same definition of flock, Gudmundsson and van Kreveld [14] showed that for any radius approximation with factor smaller than 2, computing the longest duration flock and the largest subset flock is NP-hard to compute and even NP-hard to approximate within a factor of $|T|^{1-\epsilon}$ and $|A|^{1-\epsilon}$, respectively, where T is the set of all discrete time steps, A is the set of all sensor nodes, and $\epsilon > 0$. As another such movement pattern, Andersson et al. [15] gave a generic definition of the pattern *leadership* and discussed how such leadership patterns can be computed from a group of moving entities. This work revolves around the concept of leading as "being followed but not following anyone else" for some time. Mining for such leadership patterns involves a set of algorithms operating on a set of globally preprocessed data arrays that store leading and following constellations for the investigated set of entities. All such movement patterns have in common that some geometrical relation of the involved moving objects has to persist over a certain time span.

Current data mining approaches for movement patterns rely on global knowledge and global data structures such as quadtrees, clustering, or preprocessed metadata arrays. The use of such global and hence static data structures is not well-suited for local knowledge discovery in a dynamic scenario such as a mobile wireless sensor network.

2.2 Mobile Wireless Sensor Networks

A *wireless sensor network* (WSN) is a wireless network of untethered, battery powered miniaturized computers with the ability to sense, process, and communicate information in a collaborative way [16]. A WSN monitoring a phenomenon in geographic space is called *geosensor network* [17]. The tracking of mobile targets is one typical task for geosensor networks [16, 18]. Other applications have included observing hazards [19], monitoring seismic activity [20, 21], or managing traffic flow [11]. When the sensor nodes are deployed to moving entities in a dynamic scenario, we refer to a *mobile wireless sensor network* (mWSN).

Mobility presents significant challenges to the design of mWSN applications. Maintaining static, global data structures, like connectivity graphs and routing tables, is a particular challenge in mWSN. Even in static WSN, battery power resources for wireless communication are extremely limited. With mWSN, constantly changing network topologies and sensed data means maintaining centralized global knowledge and data structures can become highly complex and inefficient.

As a result, some researchers have turned to *decentralized algorithms* (an algorithm that runs in parallel on sensor nodes without any centralized control) to address this problem. Unlike global approaches, decentralized algorithms facilitate fast local updating and do not require hard-to-maintain global consistency [22]. Other researchers including [19, 23, 24, 25] have all investigated decentralized knowledge maintenance and creation in spatial applications to cope with the problems posed by mobility in mWSN. Such decentralized approaches can also help to increase energy efficiency [16], increase scalability and robustness [26], and reduce the potential for information overload [27, 28].

As well as challenges, mobility also presents opportunities that are beginning to be exploited by some researchers. Several authors exploit the *mobility diffusion effect*, that is the diffusion of information through an mWSN from the constant hand-over of information amongst meeting moving nodes [29]. *Last encounter routing* (LER), for example, computes routes purely based on a last encounter table stored in every node. The constant rearrangement of nodes in an mWSN can furthermore be exploited to overcome unfavorable constellations. Grossglauser and Tse [22], for example, showed that for asynchronous applications with high delay tolerance, patience can increase efficiency with respect to throughput capacity when messages can wait for good routes in a network topology constantly changing due to node mobility.

In summary, the constantly changing topology amongst the moving nodes in an mWSN challenges conventional solutions that have been developed for static networks, but at the same time offers new options for in-network data processing through the exploitation of the spatiotemporal nature of movement. Decentralized spatial computing is well-suited to use in mWSN, capable of operating using purely *local* knowledge, but still aiming to monitor geographic phenomena with *global* extents [30].

3 Problem Definition

In this section, we provide a formal problem definition, including specifying the assumptions behind decentralized processing in an mWSN, and providing a precise definition of the meaning of "flock" in our scenario.

3.1 Preliminaries and Assumptions

An mWSN can be modeled as a set A of sensor nodes. The locations of sensor nodes are known for an ordered set of discrete time steps $T = \{t_1, t_2, ..., t_n\}$. The associated movement trajectories of sensor nodes in the plane ($\mathbb{R}^2$) can be modeled as a *locator* function $l : A \times T \rightarrow \mathbb{R}^2$. Thus for any time $t \in T$ and sensor node $a \in A$, $l(a, t)$ gives the coordinate location of sensor node a at time t. The distance between two sensor nodes $a, a' \in A$ at time $t \in T$ is given by the usual Euclidean metric $\delta(l(a, t), l(a', t))$. In the sequel, we write $\delta_t(a, a')$ to denote the distance between two sensor nodes a and a' at time t for conciseness, i.e., $\delta_t(a, a') := \delta(l(a, t), l(a', t))$.

At this point a few assumptions are worth noting. Without loss of generality we make the simplifying assumption that the set of sensor nodes is constant over time (i.e., that no sensor nodes leave or enter the network). We model time as a discrete domain T. This assumption also does not lead to a loss of generality, since continuous time can always be adequately approximated in a specific application with arbitrarily fine discrete time steps.

Nearby sensor nodes in an mWSN can wirelessly communicate with one another. Thus at any given point in time, each sensor node will have a (possibly empty) set of sensor nodes within its communication range, called its *neighborhood*. Given a fixed communication distance $c \in \mathbb{R}^+$, the neighborhood of a sensor node a at time t, written $nbr(a,t)$, is the set of sensor nodes within a's communication distance at time t, i.e., $nbr(a,t) := \{a' \in A | \delta_t(a,a') \leq c\}$. Even though the here described flocking scenario is scale-less, a reasonable assumption for communication range c in an implemented network of current standard sensor nodes could be 30m, allowing, for instance, the detection of flocking cattle in confinement. Note furthermore that this model assumes all sensor nodes have constant and equal communication distances. Although this is not realistic in actual sensor node networks, it is adequate for the purposes of this paper and helps simplify the formal model.

One further point worth noting is that although the locator function provides the location of each sensor node in the plane over time, our algorithm does *not* assume that individual sensor nodes have access to this information. Instead the algorithm in the next section is purely qualitative, relying instead on the neighborhoods of each sensor node. These neighborhoods emerge as a consequence of the physical characteristics of wireless communication, without the need for quantitative positioning systems such as GPS or range-finding.

3.2 Decentralized Detection of Flocks

Laube et al. [12] use the term "flocking" to describe a collective movement pattern expressed by a set of moving entities. In our context, the entities refer to the moving sensor nodes of a mWSN. A flock in this sense is any set M of n mobile sensor nodes that exhibit the same motion attribute over a given period of time of length $k \in \mathbb{R}$. Speed or movement azimuth are examples of motion attributes in this context. This initial definition is very general and does not assume any notion of proximity, which is often associated with the term flocking in common usage.

Adding a proximity constraint, [13] defined an (n,k,p)-*flock* as any set M of n mobile sensor nodes, where for every moment in a time period of k consecutive time steps, there exists some disk of pattern radius p that contains every sensor node in M. More formally and in the context of our definition of mWSN above, an (n,k,p)-*flock* is a set $M \subseteq A$ such that for every time instant $t \in \{t_i, ..., t_j\} \subseteq T$, with $j - i + 1 \geq k$, there exists a circle of radius p that spatially contains $l(m,t)$ for all $m \in M$.

The definition of an (n,k,p)-*flock* thus assumes a flock exists for a period of k consecutive time steps single. Further, the definition assumes that the flock is composed of n identical sensor nodes for its entire existence. As such, the definition is quite restrictive. Current work by the authors is looking at relaxing these constraints, and discussion of the options has already appeared in the literature [3].

The problem we address in this paper is a decentralized detection of (n,k,p)-*flocks*. Given a set of sensor nodes A, we give an algorithm that detects for any time t all present (n, k, p)-*flocks* by only relying on *local* knowledge and without any form of global information processing. As our main objective is to substitute global approaches with a purely decentralized approach, we primarily investigate the reliability of our inherently approximating decentralized solution. In this context, reliability refers to an error analysis, quantifying the number of correctly detected and missed patterns, respectively.

4 The FLAGS Algorithm

As part of this work we have developed a family of decentralized algorithms for flock detection. This section presents the FLAGS algorithm for detecting **fl**ocking **a**mongst **g**eo-**s**ensors in an mWSN. FLAGS is designed to find flock patterns without central control, solely by decentralized collaboration of roaming sensor nodes, that only perceive their immediate neighbors via simple "hello" messages.

The algorithm is distributed as it runs in parallel on all sensor nodes at the same time. The algorithm is decentralized in the sense that each sensor node a can only access information about its immediate neighborhood, specifically the identity of sensor nodes that are within direct one-hop communication range of a at time t, i.e., $nbr(a, t)$. The algorithm is dynamic, because each sensor node is constructing new information about the existence of flocks on-the-fly as it moves through time and space.

4.1 Handing Around Maturing Information Tokens

In the FLAGS algorithm, the collective of roaming sensor nodes administers information tokens that accumulate knowledge about flock patterns over time. A token is simply a pair (X, j), where X is a set of sensor nodes that a knows currently lie within a circle of radius p during the previous j time steps. The set of tokens stored by a sensor node a at time t is denoted $\mathit{TokenSet}(a, t)$.

At each time step t_i the algorithm works as follows (see Algorithm 1): Each sensor node a computes its set of neighbors $nbr(a, t_i)$ by sending and receiving simple "hello" messages (line 1). If as current set of neighbors contains at least as many sensor nodes as are required for detecting a flock (denoted by the threshold ν, related to the number of nodes in the flock n and discussed further in the next section), a will create a new token containing this information and add it to $\mathit{TokenSet}(a, t_i)$ (lines 3–4). Initially, $\mathit{TokenSet}(a, t_i)$ is empty (line 2). Next, a will update and check all its tokens $(X, j) \in \mathit{TokenSet}(a, t_{i-1})$ from the previous time step t_{i-1} (line 5). The token (X, j) is updated by $X := X \cap nbr(a, t_i)$ to indicate that now only a smaller set of sensor nodes are in a's neighborhood, and by $j := j + 1$ to reflect the increased number of time steps that the sensor nodes in the updated X have been neighbors of a (line 6). If the size of the updated X is at least the number of sensor nodes required for detecting a flock ν, the token is added to $\mathit{TokenSet}(a, t_i)$ (lines 7–8). After that, the sensor node a inspects all tokens in $\mathit{TokenSet}(a, t_i)$, and whenever it finds a token of age $j \geq k$ it triggers a "pattern found" message (lines 9–11). As a last but very important step, sensor node a will exchange its set of tokens with its neighbors (line 12). Figure 2 illustrates this procedure and Algorithm 1 formalizes it.

FLAGS decouples knowledge about patterns from sensor nodes. Knowledge diffuses through the network in form of knowledge collecting tokens that are handed over between roaming sensor nodes. Even if all members of a flock rearrange at each consecutive step, the spatiotemporal knowledge describing that flock is very likely to persist, because it is likely that at least one sensor node holds an appropriate token.

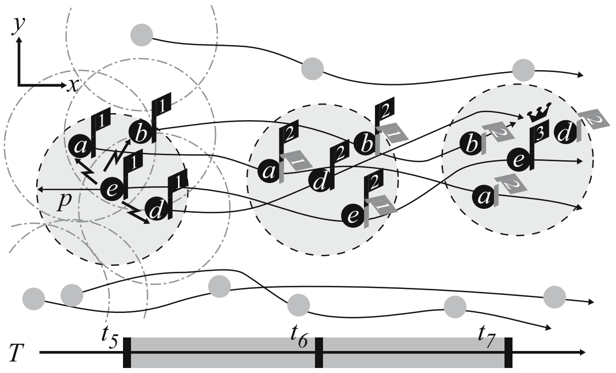


Fig. 2. Maturing knowledge tokens detect a $(n = 4, k = 3, p)$-*flock* locally when $c \approx p$. At t_5 sensor node e counts the required 4 neighbors, creates a token $(\{a, b, d, e\}, 1)$ that is transmitted to all its neighbors within communication range c (little numbered flags). At t_6, after having moved and potentially rearranged, all sensor nodes check their tokens. This time only sensor node d counts enough neighbors and ages his token to $(\{a, b, d, e\}, 2)$. All other sensor nodes drop their token. Sensor node d, however, forwards its aged token to all its neighbors. Finally, at t_7, again e counts enough neighbors, its token $(\{a, b, d, e\}, 3)$ reaches the mature age $k = 3$, and flags a "found pattern"-message (crown).

4.2 Local Extrapolation

The description of the FLAGS algorithm in the previous section was deliberately vague about the precise value of ν, the "number of sensor nodes required for detecting a flock." This section explains how a correct value for ν is chosen in the FLAGS algorithm.

As discussed above, the spatial awareness of the roaming sensor nodes is constrained by their neighborhood, which in turn is constrained by their communication range c. As a result, the ratio of the sensor node's communication range c and the pattern radius p, informally termed a node's *perception range*, plays a critical role in detecting flocks in a decentralized algorithm. Figure 3 illustrates the $\frac{c}{p}$-ratio for $c \ll p$, $c \approx p$, and $c \gg p$, respectively.

Constellations $c \gg p$ are of limited interest from a decentralized spatial computing perspective, since in these cases there is every chance a single sensor node will locally be able to observe the entire flock (ultimately converging toward

Algorithm 1. Procedure that is locally run once at each time step t_i in each sensor node a for a global value ν.

```
    // initialization
1   communicate "hello" messages and construct set of neighbors $nbr(a, t_i)$
2   $TokenSet(a, t_i) \leftarrow \emptyset$
    // create new token if the neighborhood is large enough
3   if $|nbr(a, t_i)| \geq \nu$ then
4       $TokenSet(a, t_i) \leftarrow TokenSet(a, t_i) \cup \{(nbr(a, t_i), 1)\}$
    // check tokens of previous time step; if valid, update and age them
5   foreach $(X, j) \in TokenSet(a, t_{i-1})$ do
6       $X_{new} \leftarrow X \cap nbr(a, t_i)$
7       if $|X_{new}| \geq \nu$ then
8           $TokenSet(a, t_i) \leftarrow TokenSet(a, t_i) \cup \{(X_{new}, j + 1)\}$
    // check for patterns in current set of tokens
9   foreach $(X, j) \in TokenSet(a, t_i)$ do
10      if $j \geq k$ then
11          report "pattern found"
    // information diffusion
12  communicate with neighbors and update $TokenSet(a, t_i)$
```

sensor nodes having global knowledge). By contrast, $c \approx p$, and particularly $c \ll p$ require strategies and heuristics allowing sensor nodes to overcome their own limited perception range.

To correct for limited communication range, FLAGS applies a local extrapolation heuristic, by assuming that a sensor node that is part of a flock can expect a neighborhood size that is in proportion to its perception range. In other words, a sensor node for which $c \ll p$ detects a flock constellation when its neighborhood contains approximately $\nu = \frac{c^2}{p^2} * n$ sensor nodes. In Fig. 3(a) the central sensor node has a neighborhood of 5 (including itself) and hence might infer

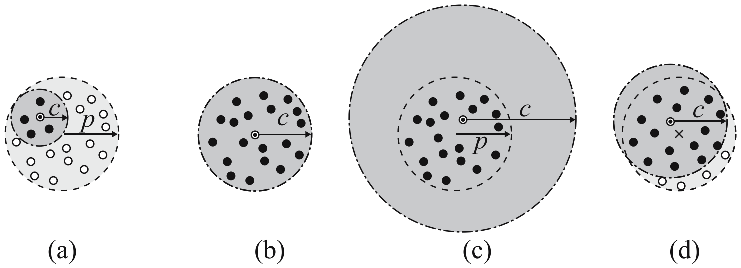


Fig. 3. The $\frac{c}{p}$-ratio rules the difficulty of detecting flock patterns in a decentralized way, illustrated are the ratios $c = 0.5p$ (a), $c = 1p$ (b), and $c = 2p$ (c) respectively. Furthermore, even if $c = 1p$ there is no guarantee for "seeing" all sensor nodes of a flock, as the most central sensor node may be eccentric (d).

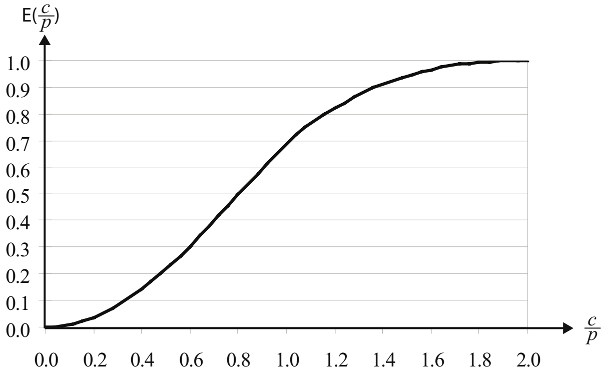


Fig. 4. Compensation factor E over $\frac{c}{p}$. For $c = 0.5p$ we get $E(0.5) = 0.22$, for $c = 1p$ the compensation factor is $E(1.0) = 0.69$, and for $c = 2p$ it is $E(2.0) = 1.0$.

the presence of a flock of size $n = 20$ having sensed the required fraction (in this case: $\frac{1}{4}$) of n. Obviously, local extrapolation assumes that the density of sensor nodes within communication range c approximates the density of sensor nodes in the flock. In many cases this is a fair assumption. However, if the sensor nodes cluster in only one segment of the flock area and hence some individual nodes in the sparsely populated segments miss out on that flock, the flock as a whole would be detected by the sensor nodes in the dense segment instead.

However, this first approximation fails as it assumes that some sensor node is located at the exact center of the flock. Any eccentricity in the most central node's location means a sensor node's communication area is expected to only cover a smaller part of the flock area, shown in Fig. 3(d). As a result, we expect a sensor node in a flock to sense on average slightly fewer than $\frac{c^2}{p^2} * n$ neighbors due to its eccentricity with respect to the center of the flock.

To correct for this effect a compensation factor E was introduced. E was computed as the expected fraction of the flock area (a circle of radius p) that is covered by the communication area (a circle of radius c, with center inside the flock area) of a uniformly randomly placed sensor node inside this flock. The formal mathematical details of the derivation of E are outside the scope of this paper. Instead we provide a short explanation of our approach in the following paragraph and an illustrative Fig. 4 plotting E over various $\frac{c}{p}$ ratios .

To compute E, we model the flock and communication areas as two disks in the plane with radius p and c, respectively. The distance between the centers of the disks is denoted by s. These disks have nonempty intersection, since $s \leq p$. We computed the area of intersection as a function A of p, c and s. We then defined a function R of p, c and s, as the ratio of the area A and the flock area. The expected value of this function for a uniformly randomly chosen s with $0 \leq s \leq p$, is derived from the integral of R from $s = 0$ to $s = p$. This yields a function E of p and c. But since R and E are defined to be ratios, E is only a function of $\frac{c}{p}$ and not the actual values of p and c.

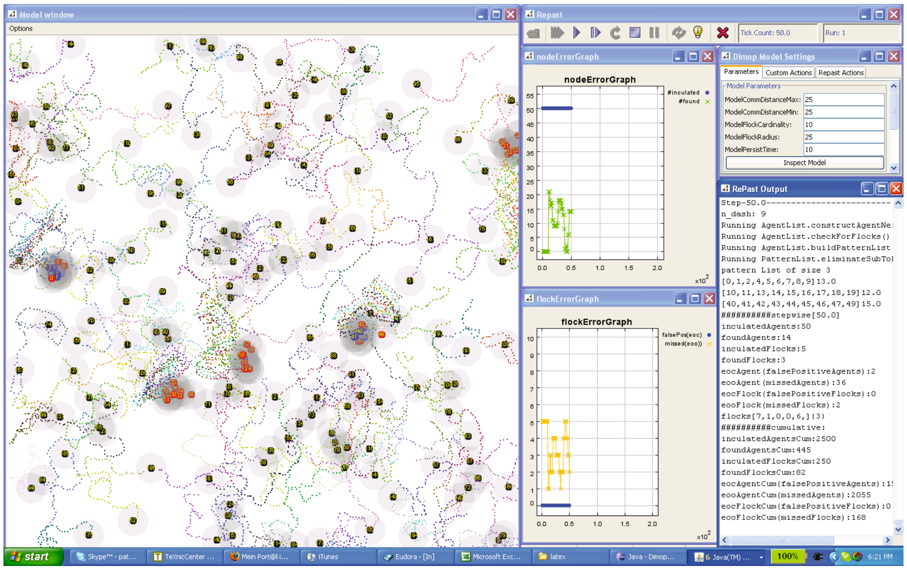
In the FLAGS algorithm, the threshold ν is used as the number of sensor nodes required for detecting a flock, defined as follows:

$$\nu := E\left(\frac{c}{p}\right) \cdot n$$

The effect of the compensation factor E is to reduce the number of neighbors a sensor node that is part of a flock expects to have. As discussed above, this reduction allows for the likelihood that, on average, sensor nodes are not ideally situated to perceive the best possible number of neighbors.

5 Evaluation

This section reports on implementation and testing of the FLAGS algorithm. In addition to testing the reliability of the algorithm, this section also investigates to what degree the underlying movement regime influences the performance of our decentralized knowledge discovery algorithm. For evaluation purposes, a simulation environment for detecting flocks in given trajectories was implemented using the popular open source agent-based modeling toolkit REPAST (see Fig. 5).



ModelCommDistanceMax:	25
ModelCommDistanceMin:	25
ModelFlockCardinality:	10
ModelFlockRadius:	25
ModelPersistTime:	10

```
Step-50.0------------------------
n_dash: 9
Running AgentList.constructAgentNe
Running AgentList.checkForFlocks()
Running AgentList.buildPatternList
Running PatternList.eliminateSubTo
pattern List of size 3
[0,1,2,4,5,6,7,8,9]13.0
[10,11,13,14,15,16,17,18,19]12.0
[40,41,42,43,44,45,46,47,49]15.0
##########stepwise[50.0]
inculatedAgents:50
foundAgents:14
inculatedFlocks:5
foundFlocks:3
eocAgent(falsePositiveAgents):2
eooAgent(missedAgents):36
eocFlock(falsePositiveFlocks):0
eooFlock(missedFlocks):2
flocks[7,1,0,0,6,](3)
##########cumulative:
inculatedAgentsCum:2500
foundAgentsCum:445
inculatedFlocksCum:250
foundFlocksCum:82
eocAgentCum(falsePositiveAgents):1
eooAgentCum(missedAgents):2055
eocFlockCum(falsePositiveFlocks):0
eooFlockCum(missedFlocks):168
```

Fig. 5. FLAGS-implementation in REPAST. The framework features a map view (communication ranges as gray shades, trajectories as dotted lines), two error plots (*eoo*, *eoc*), a parameter and a log window.

5.1 Data

Simulated trajectory data was used instead of real movement observation data, as this allows control of both the movement regime and the flocking behavior of the moving sensor nodes in the experiments. Different movement regimes were expected to have a significant influence on the performance of the FLAGS algorithm, so simulated trajectories that follow well-known movement regimes were used: unconstrained random movement (URM) and correlated random walk (CRW). Using simulated movement observation data also had the experimental advantage that the number of flocks present in the data could be easily controlled. This was crucial for the performance analysis, which evaluates how many of the implanted patterns are actually found by the FLAGS algorithm.

For all experiments a population A of sensor nodes ($|A| = 200$) was generated moving over $|T| = 100$ discrete time steps in an underlying square space of size $8192 * 8192$ positions. 50 out of 200 sensor nodes were given flocking behavior, implanted in five flocks consisting of $n = 10$ flocking sensor nodes each with a pattern radius $p = 250$. The flocking behavior follows the definition of *flock* outlined in Sect. 3.2.

Unconstrained Random Movement (URM). In order to maximize the comparability of our decentralized approach with previous work, the first set of experiments used the same unconstrained random movement (URM) generator as [13]. In the URM data sets, the location of a sensor node is completely random, and unrelated to any previous steps. Non-flocking sensor nodes randomly "jump" around in simulation space. Flocking sensor nodes, however, are at each time step randomly positioned withing a circle of size pattern radius $p = 250$, itself randomly positioned.

Correlated Random Walk (CRW). Initial experiments showed that URM only allowed for very limited exploitation of the spatiotemporal characteristics of movement. Anticipating that any decentralized pattern detection approach would rely in part on exactly such exploitation, a second set of experiments was conducted with trajectories generated using correlated random walks.

In a random walk, the location of a sensor node in successive timesteps is determined by a randomized displacement from the previous time step. In correlated random walk (CRW) steps of sensor nodes are correlated by making the direction and/or step length of the current move bias the direction and step length of the next move [31]. Step length and turning angle of the subsequent move are typically drawn from some stochastic frequency distribution, for instance in the case of direction with turning angles concentrated around $\mu = 0$. CRW provides a more realistic model of random spatial movement than URM, as consecutive locations of sensor nodes are highly correlated in space.

In the experiments a normal distribution with a direction change $N(\mu = 0, \sigma = 0.6 * \pi)$ and a step length of $N(\mu = 50, \sigma = 15)$ was used. Each sensor node walked a trajectory of approximate length $100*50 = 5000$ units. In addition to the CRW property, flocking sensor nodes satisfied the flock property, by simply ensuring that they were aggregated within the given pattern radius $p = 250$. The map view in Fig. 5 illustrates first 50 time steps of a CRW data set.

5.2 Experiments

Following the general WSN constraint that communication consumes more energy than sensing, algorithm efficiency often bases on keeping the number of messages low. For the evaluation of the here discussed decentralized data mining application, a slightly different perspective was taken. It was assumed that our sensor nodes constantly know in a qualitative manner by which neighbors they are surrounded. Since it had to be anticipated that the decentralized data mining algorithm is rather an approximation than an exact solution, performance was evaluated as the ratio of patterns found with the centralized and the decentralized approach. For the sake of simplicity, this evaluation focused on the pattern detection and did not investigate the details of routing the information back to the query source once the patterns had been detected.

FLAGS reliability was tested for both URM data and CRW data (Fig. 6). In both cases two levels of flock contiguity were tested: In order to be detected flocks were required to be detected for $k = 3$, and $k = 10$ consecutive time steps out of $|T| = 100$ time steps in total. Note that each flock could only be counted once, even if several sensor nodes would find the same flock. For a total of 5 simulation runs and four combinations each, the *detection error* for a variable $\frac{c}{p} - ratio$ was computed.

– *detection error* (y-axis): At each time step the number of found patterns was compared with the known number of implanted patterns. Errors of omission (*eoo*) and errors of commission (*eoc*) were evaluated and averaged over all $|T| = 100$ time steps[1].
  - *eoo* ("missed patterns"): As 5 flocks consisting of 10 sensor nodes each were implanted, over $|T| = 100$ time steps a total 500 flocks could be detected correctly. An experiment run missing 100 flocks would result in $eoo = 0.2$.
  - *eoc* ("false positives"): FLAGS uses heuristics in order to infer the presence of patterns (local extrapolation and compensation factor E, see Sect. 4.2). Hence, FLAGS may detect patterns where there are no patterns to be found. Such false positives could, for example, emerge when $c \ll p$ and $\nu = 2$, as it might happen that two independent sensor nodes randomly intermingle for a long enough time. All such false positives were counted and again averaged over the whole experiment run. Again, with a total of 500 flocks per run, a total of 50 false positives would result in $eoc = 0.1$.

– $\frac{c}{p} - ratio$ (x-axis): FLAGS performance was evaluated for a variable $\frac{c}{p} - ratio$. Given a pattern radius $p = 250$ the communication range c was varied in the interval of $[0, \ldots, 500]$, hence in a $\frac{c}{p} - ratio$ interval of $[0, \ldots, 2]$.

[1] Please note that in fact the error was averaged over $|T| - k + 1$ time steps, as in the first $k - 1$ time steps no flocks can be detected due to their temporal extent.

Obviously, the above evaluation experiments do not account for flocks that may randomly emerge in the generated data set, but were not purposely implanted in the first place. However, the emergence of such random flocks is minimized by a relatively large flock size $n = 10$, and gets very unlikely with increasing flock contiguity k. Furthermore, such random flocks would only influence *eoc*, as in fact existing random flocks could be misinterpreted as false positives. Hence, our evaluation, in the worst case, overestimates *eoc*.

5.3 Results

All four graphs show a few similar properties (Fig. 6). No results are provided when $\frac{c}{p} < 0.48$, as below that threshold ν drops below 2 and hence pattern detection is not possible anymore (see Fig. 4). Furthermore, the error graphs often show "zigzagging" shape. This feature emerges since the number of counted neighbors inherently changes in discrete steps and hence FLAGS performance alters in grades.

URM, $k = 3$. As long as the communication range c is slightly larger than the pattern radius p, no patterns are missed out and *eoc* is 0. Around $\frac{c}{p} \approx 1.0$ a first *eoo*-peak emerges when large numbers ν are required for token generation. Below 1.0, the performance decreases gradually, as flock sensor nodes randomly shuffled in the flock disk result in more and more misses. No *eoc* was recorded. Given that the sensor nodes relocate randomly at each step, it is very unlikely that a significant number of sensor nodes satisfy the flock definition randomly.

URM, $k = 10$. As with $k = 3$, *eoo* emerges at $\frac{c}{p} < 1.2$. However, with $k = 10$ it is almost impossible that any sensor node re-finds the very *same* neighbors out of the $n = 10$ neighbors, hence FLAGS' performance degrades rapidly to total failure shortly below 0.8. Again, and for the very same reasons as above, no *eoc* was recorded.

CRW, $k = 3$. In the CRW experiment FLAGS performs significantly better than with URM, for $k = 3$ the performance is almost flawless. A t-test for 5 simulation runs indicates that for any ratio $\frac{c}{p} < 0.84$, *eoo* for CRW is significantly lower than for URM on the 95% significance levels. The explanation for this good performance lies in the locality property of CRW. Even when c drops below p, subgroups within a flock tend to stay together and hence knowledge about subgroups persists, allowing the consistent extrapolation of pattern knowledge using the compensation factor $E(\frac{c}{p})$. Also *eoc* shows moderate levels and only expresses noticeable values when ν reaches 2 at $\frac{c}{p} = 0.56$. The random event of two nodes staying together for $k = 3$ time steps is moderately likely with the chosen parameters and this *eoc* peak is hence unsurprising.

CRW, $k = 10$. Again, *eoo* is moderate, but admittedly on a higher level (according to a t-test significantly higher at the 95% level for $0.78 < \frac{c}{p} < 1.12$). The constraint of $k = 10$ makes it harder to maintain knowledge about subgroups. On a positive note, the *eoc* does not peak at all as it is just too unlikely that random patterns emerge with even $\nu = 2$.

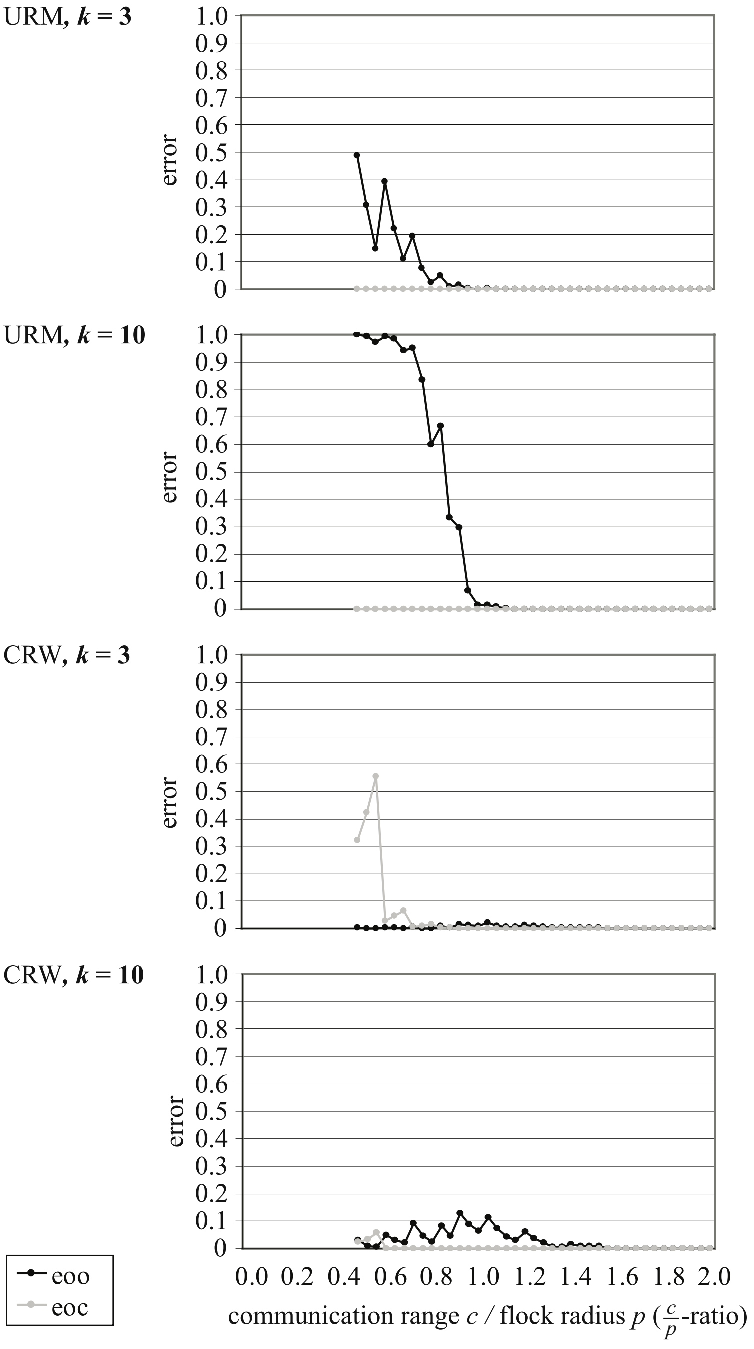


Fig. 6. Error analysis for URM and CRW experiments

6 Discussion

Three major findings emerge from our experiments. First, with the used parameters, *eoc* is not an issue. Apart from the single peak with small c for $k = 3$, *eoc* is negligible. This can be explained that with the used parameters even for small νs and short ks it is very unlikely that enough sensor nodes stay together for enough time steps. Different flock parameters and different densities amongst non-flocking sensor nodes may admittedly result in higher *eoc* values. Second, longer contiguities k are understandably harder to detect as a longer time required for tokens to mature opens up more possibilities for failure. Third, FLAGS performs much better for CRW than for URM, as we would expect from an algorithm developed for detecting structure in structured movement data. Since FLAGS collects neighborhood knowledge over several consecutive time steps, locality where individuals tend to stay together for some time, obviously helps for the crucial token maturing. Even though we assumed in our experiments random movement, we could show that FLAGS performs better as movement becomes less random. We hence argue that the FLAGS algorithm is taking advantage of the opportunity of movement rather than purely dealing with the challenges of movement.

This last finding agrees with similar experiences in the LER experiments in [29]. In their experiments the performance of their history based routing algorithm depended heavily on the nature of the mobility regime and the parameterization of the experimental setup. LER succeeds only since the sensor nodes perform random walks and thus sensor node mobility diffuses estimates of the destination's location sufficiently quickly and densely throughout the dynamic mWSN. Even though the preferred regime was random walk in their case and correlated random walk in ours, it seems apparent that the inevitably very specialized algorithms for decentralized spatial computing operate best within the constraints that they have been designed for.

Our initial experiments with decentralized data mining suggest that we simply have to accept the fact that any decentralized solutions inferring global knowledge from partial and potentially imperfect local information must allow for approximation solutions. One way of giving the limited sensor nodes the leeway to do so is the use of heuristics, as applied with the local extrapolation approach and the compensation factors E in FLAGS. We allow our sensor nodes to detect and report patterns even if they actually only found fractions of patterns. For many decentralized data mining applications we will have to allow for some error in order to achieve a workable solution at all. For the parameterizations used in our experiments, the heuristics provided very useful tools without introducing too much of *eoc*. In other cases one might have to accept some *eoc* in order to keep *eoo* low. Obviously, decentralized approximation solutions could always be used as a cost-effective preliminary data mining step that might trigger more reliable but also more expensive approaches.

FLAGS exploits the spatiotemporal properties of movement as it features information exchange amongst roaming neighbors. This constant process of exchanging and validating information allows individual sensor nodes to learn from their neighbors about processes beyond their own limited perception range. However, recording of the past and hence "memory" is purely outsourced to the knowledge tokens, the sensor nodes themselves are oblivious. Clearly, alternative approaches can be thought off where individual sensor nodes, other than exchange their knowledge with their neighbors, also record what they see along their trajectory, and integrate such spatiotemporal knowledge in order to gain the big picture beyond their limited perception range.

As mentioned earlier, we use the same definition of flock and the same data generator (for URM trajectories) as in [13]. However, a direct comparison between the two studies is not meaningful, because in [13], the results focus on theoretical and experimental running times, while the evaluation of our approach bases on an error analysis, investigating on the reliability of a decentralized solution.

7 Conclusions

In this paper we revisited the generic movement pattern "flock" and showed that such patterns formed by individuals are very suitable for decentralized and hence "ego-centric" pattern detection algorithms. We introduced FLAGS (**fl**ocking **a**mongst **g**eo-**s**ensors) a solely decentralized flock detection algorithm. FLAGS exploits the spatiotemporal properties of movement as roaming sensor nodes exchange information tokens that collect knowledge about patterns. Experiments with simulated movement data show that FLAGS successfully completes its task without central control when *a priori* knowledge about the movement regime can be exploited.

As a first general lesson learned, we argue that separating knowledge from sensor nodes (in our specific case in the form of "floating knowledge tokens"), is a promising strategy for overcoming the limited spatial perception of individual sensor nodes in an mWSN. Second, we acknowledge that in a decentralized setting heuristics are a suitable way for compensating for the limited perception of individual sensor nodes. We thirdly identify the need to further explore locality. Hence, currently we are working on improved versions of FLAGS that use a memory function. Sensor nodes with memory are enabled to "graze" information in space-time and hopefully assemble spatially limited glimpses in order to form the bigger picture.

Acknowledgements

Patrick Laube is funded by the Australian Research Council (ARC), Discovery grant DP0662906 and the ARC Research Network on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP). Matt Duckham's research is supported by the Australian Research Council under ARC Discovery Grant DP0662906. Matt is also grateful for the support of the Ordnance Survey of Great Britain. Thomas Wolle's work is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council. Thomas and Patrick also acknowledge the GADGET Workshop on

Geometric Algorithms and Spatial Data Mining, funded by the Netherlands Organisation for Scientific Research (NWO) under BRICKS/Focus grant number 642.065.503. Finally, all authors wish to thank three anonymous reviewers for their valuable comments.

References

[1] Galton, A.: Dynamic collectives and their collective dynamics. In: Cohn, A., Mark, D.M. (eds.) COSIT 2005. LNCS, vol. 3693, pp. 300–315. Springer, Heidelberg (2005)

[2] Gudmundsson, J., Laube, P., Wolle, T.: Movement patterns in spatio-temporal data. In: Shekhar, S., Xiong, H. (eds.) Encyclopedia of GIS, pp. 726–732. Springer, Heidelberg (2008)

[3] Gudmundsson, J., van Kreveld, M., Speckmann, B.: Efficient detection of patterns in 2D trajectories of moving points. GeoInformatica 11(2), 195–215 (2007)

[4] Andersson, M., Gudmundsson, J., Laube, P., Wolle, T.: Reporting leaders and followers among trajectories of moving point objects. GeoInformatica (in press)

[5] Shirabe, T.: Correlation analysis of discrete motions. In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 370–382. Springer, Heidelberg (2006)

[6] Laube, P., Imfeld, S., Weibel, R.: Discovering relative motion patterns in groups of moving point objects. International Journal of Geographical Information Science 19(6), 639–668 (2005)

[7] Batty, M., Desyllas, J., Duxbury, E.: The discrete dynamics of small-scale spatial events: agent-based models of mobility in carnivals and street parades. International Journal of Geographical Information Science 17(7), 673–697 (2003)

[8] Wark, T., Corke, P., Sikka, P., Klingbeil, L., Guo, Y., Crossman, C., Valencia, P., Swain, D., Bishop-Hurley, G.: Transforming agriculture through pervasive wireless sensor networks. Pervasive Computing, IEEE 6(2), 50–57 (2007)

[9] Russell, M.: Attention all units: Dick tracy is on its way. The Sunday Age, 9 (2007)

[10] Correll, N., Martinoli, A.: Collective inspection of regular structures using a swarm of miniature robots. In: Ang, J., Marcelo, H., Khatib, O. (eds.) Experimental Robotics IX, The 9th International Symposium on Experimental Robotics (ISER), Singapore, June 18-21. Springer Tracts in Advanced Robotics, pp. 375–385. Springer, Heidelberg (2006)

[11] Kellerer, W., Bettstetter, C., Schwingenschlogl, C., Sties, P., Steinberg, K.E. (auto) mobile communication in a heterogeneous and converged world. IEEE Personal Communications 8(6), 41–47 (2001)

[12] Laube, P., van Kreveld, M., Imfeld, S.: Finding remo - detecting relative motion patterns in geospatial lifelines. In: Fisher, P.F. (ed.) Developments in Spatial Data Handling. Proceedings of the 11th International Symposium on Spatial Data Handling, pp. 201–214. Springer, Berlin (2004)

[13] Benkert, M., Gudmundsson, J., Hübner, F., Wolle, T.: Reporting flock patterns. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 660–671. Springer, Heidelberg (2006)

[14] Gudmundsson, J., van Kreveld, M.: Computing Longest Duration Flocks In Trajectory Data. In: GIS 2006: Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems, Arlington, Virginia, USA, pp. 35–42. ACM, New York (2006)

[15] Andersson, M., Gudmundsson, J., Laube, P., Wolle, T.: Reporting leadership patterns among trajectories. In: 22th Annual ACM Symposium on Applied Computing, Seoul, Korea, pp. 3–7 (2007)
[16] Zhao, F., Guibas, L.J.: Wireless Sensor Networks – An Information Processing Approach. Morgan Kaufmann Publishers, San Francisco (2004)
[17] Nittel, S., Stefanidis, A., Cruz, I., Egenhofer, M.J., Goldin, D., Howard, A., Labrinidis, A., Madden, S., Voisard, A., Worboys, M.: Report from the first workshop on geo sensor networks. ACM SIGMOD Record 33(1) (2004)
[18] Guibas, L.J.: Sensing, tracking and reasoning with relations. Signal Processing Magazine, IEEE 19(2), 73–85 (2002)
[19] Duckham, M., Nittel, S., Worboys, M.: Monitoring dynamic spatial fields using responsive geosensor networks. In: 13th annual ACM international workshop on Geographic Information Systems, Bremen, Germany, pp. 51–60. ACM Press, New York (2005)
[20] Werner-Allen, G., Johnson, J., Ruiz, M., Lees, J., Welsh, M.: Monitoring volcanic eruptions with a wireless sensor network. In: Second European Workshop on Wireless Sensor Networks, pp. 108–120 (2005)
[21] Werner-Allen, G., Lorinez, K., Ruiz, M., Marcillo, O., Johnson, J., Lees, J., Welsh, M.: Deploying a wireless sensor network on an active volcano. IEEE Internet Computing 10(2), 18–25 (2006)
[22] Grossglauser, M., Tse, D.N.C.: Mobility increases the capacity of ad hoc wireless networks. IEEE-ACM Transactions on Networking 10(4), 477–486 (2002)
[23] Wolfson, O., Ouksel, A., Xu, B.: Resource discovery in disconnected mobile ad-hoc networks. In: Proc. International Workshop on Next Generation Geospatial Information (2003)
[24] Wolfson, O., Xu, B., Sistla, A.P.: An economic model for resource exchange in mobile peer to peer networks. In: Proc. 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004) (2004)
[25] Wu, Y.H., Guan, L.J., Winter, S.: Peer-to-peer shared ride systems. In: Nittel, S., Labrinidis, A., Stefanidis, A. (eds.) Advances in Geosensor Networks. LNCS, vol. 4540. Springer, Berlin (2007)
[26] Estrin, D., Govindan, R., Heidemann, J., Kumar, S.: Next century challenges: scalable coordination in sensor networks. In: MobiCom 1999: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, Seattle, Washington, United States, pp. 263–270. ACM, New York (1999)
[27] Rabiner, H.W., Kulik, J., Balakrishnan, H.: Adaptive protocols for information dissemination in wireless sensor networks. In: 5th annual ACM/IEEE international conference on Mobile computing and networking, Seattle, Washington, United States, pp. 174–185. ACM, New York (1999)
[28] Datta, S., Bhaduri, K., Giannella, C., Kargupta, H., Wolff, R.: Distributed data mining in peer-to-peer networks. IEEE Internet Computing 10(4), 18–26 (2006)
[29] Grossglauser, M., Vetterli, M.: Locating mobile nodes with ease: learning efficient routes from encounter histories alone. IEEE/ACM Transactions on Networking 14(3), 457–469 (2006)
[30] Estrin, D., Govindan, R., Heidemann, J.: Embedding the internet - introduction. Communications of the ACM 43(5), 38–41 (2000)
[31] Turchin, P.: Quantitative Analysis of Movement: Measuring and Modelling Population Redistribution in Animals and Plants. Sinauer Publishers, Sunderland (1998)

A Framework for Sensitivity Analysis in Spatial Multiple Criteria Evaluation

Arika Ligmann-Zielinska[1] and Piotr Jankowski[2]

[1] Department of Geography
Michigan State University
116 Geography Building
East Lansing, MI 48824-1117, USA
arikalz@gmail.com

[2] Department of Geography
San Diego State University
5500 Campanile Drive
San Diego, CA 92182-4493, USA
piotr@geography.sdsu.edu

Abstract. The paper presents a framework for sensitivity analysis (SA) in spatial multiple criteria evaluation (S-MCE). The framework focuses on three aspects of S-MCE: spatiality, scope, and cardinality. Spatiality stresses the importance of spatial criteria and spatial weights that should be explicitly considered in GIS-based MCE. Scope relates to the extent of SA, ranging from local one-at-a-time criterion examination to global testing of interdependencies among the multiple criteria model components. Cardinality addresses the duality of motivation for performing SA, namely, single-user learning and group consensus building. The framework organizes the existing SA techniques according to spatiality and scope and can be used as a conceptual guide in selecting SA techniques fitting a task at hand.

Keywords: Spatial Multiple Criteria Evaluation, Sensitivity Analysis, SDSS.

1 Introduction

Spatial multiple criteria evaluation (S-MCE) belongs to one of the basic analytical methods of GIS [16]. Over the last two decades, much work has gone into integrating MCE techniques with GIS to support formulation, modeling, and evaluation of spatial decision problems [9,17,28,33,39,40,54]. S-MCE is a method supporting a rational decision process in which a set of geographical options is evaluated based on a number of decision criteria in search of the best choice [39]. Many applications of spatial decision support systems (SDSS), of which S-MCE is a part, suggest that S-MCE has become a well established procedure for solving spatial choice problems. However, S-MCE models have also been criticized for the inadequate treatment of uncertainty present in model outcomes. For example Uran and Janseen [56], in their assessment of five SDSS models, identify the shortcomings of spatial post-processing analysis of option rankings generated with S-MCE. The uncertainty arises more often than not from the preliminary character of data and unstable human preferences. We may argue that the potential of using S-MCE lies in its exploratory approach to analyzing the decision problem [40]. Thus, ironically, the very essence of S-MCE might be at the same time its weakest point.

In this paper, we argue for strengthening the exploratory role of S-MCE by focusing on sensitivity analysis (SA) as part of the decision support methodology. In response to the lack of systematic treatment of SA in the S-MCE literature, we propose a framework for organizing and guiding the use of SA techniques. According to the framework, the objective of SA in S-MCE is to strengthen the confidence in the obtained solution or, in the case of weak confidence, help to redefine the set of acceptable alternative solutions. In the proposed framework, we extend SA beyond the criterion data uncertainty to account for the spatial characteristics of a geographic decision situation.

The roots of formal SA may be traced to engineering and scientific predictive modeling [19,22,44]. Its role has been also recognized in decision sciences, where the main purpose of SA is strengthening the bases for a decision recommendation. In Simon's decision process framework [51] of intelligence-design-choice, SA is perceived as the core of the final *choice* stage, where the decision maker evaluates and selects the desirable solution [18,28,39]. Still, a comprehensive SA included in S-MCE models is more the exception than the norm [14].

The importance of SA may be attributed to the complexity of decision processes dealing with spatial choice. Complex spatial problems involve irreducible (aleatory) uncertainty [24] caused by the difficulty of arriving at a stable preference structure for decision makers. Aleatory uncertainty is a result of semi- or ill-structured decision problems, where the decision makers are unable to define fully the problem [4,15,40]. A semi-structured decision problem may involve, for example, an incomplete or vague knowledge of decision option impacts, in which case SA could be used to examine the sensitivity of evaluation results derived from various plausible impact characteristics.

The spatial nature of geographic problems amplifies the complexity of decision making through spatial interdependencies and spatio-temporal dynamics. Future impacts of proposed choices are often stochastic and deeply uncertain [34]. The solutions to such problems should be thoroughly evaluated to ensure their robustness under a wide range of possible conditions [35,39]. Unlike the optimal outcomes, which are based on normative computational tools, there is a need to look for robust solutions in the presence of a broad spectrum of beliefs and values and under varying future conditions. Robustness is defined here as the minimal response of a model solution caused by changing input conditions. In particular, robust S-MCE solutions are characterized by rank-order stability [31], where the prioritization of options is not significantly affected by minor changes in evaluation components [3].

Traditionally, SA has been defined as the analysis of response of a model to changes in input parameters [32,37,39,57]. Usually, the general question asked in

SA is: *given the outcome, what is its sensitivity to changes in initial conditions?* Therefore, in order to perform S-MCE SA, the decision maker must have a ranking of options already in hand. In this paper, we conceptualize SA more broadly. We define SA in S-MCE as a thought experiment [1] or computer-assisted reasoning [34] aimed at quantitative and qualitative assessment of the stability of a given option ranking. This definition is not restricted to the analysis of the stability of ranking given changes in input parameters. Such a strict definition is counter to the uncertain nature of spatial decision making. An overly structured SA does not fit into an often substantially unstructured spatial problem [18]. Hence, we should account for various other - intangible and qualitative - decision factors that may influence the choice. Furthermore, the spatially-explicit nature of geographical problems calls for new, spatially-explicit methods of SA [18]. The ranking of alternatives should be analyzed based on both site-specific criterion outcomes and on spatial relations, such as proximity, contiguity, or clustering.

Finally, we should distinguish between sensitivity analysis and uncertainty analysis (UA). According to Saltelli et al. [47] UA is forward-looking in nature. Therefore, performing UA, assumptions are mapped onto inferences [46], whereas in SA a backward-looking or reverse analysis is undertaken. UA embraces multiple solutions and does not refer to any specific (initial) solution. SA focuses on the robustness of a specific solution. This paper addresses the theoretical and methodological foundations of SA in S-MCE.

In the following section we present a framework for SA in S-MCE, followed by a review of SA methods. In the final section, we outline research challenges and recommend directions for further research on SA in S-MCE.

2 A Framework of Sensitivity Analysis in Spatial Multiple Criteria Evaluation

The goal of the proposed framework is to provide an organizational outline of SA including many methods and multiple analysis pathways. We start from a three-dimensional conceptualization of SA, followed by a methodology and a review of techniques that may be used to uphold the scope and spatiality perspectives. The following questions guided the formulation of the framework:

1. Which elements of the decision process are the least informational and thus especially suited for identification in the course of SA? Are they spatially sensitive? What types of measures should be used to analyze the sensitivity of these decision components?
2. What is the informational extent of SA? Does it embrace 'the big picture of the decision problem' or 'a more focused exploration'?
3. What is the motivation for performing SA? Is it individual knowledge discovery or group consensus building?

To address these questions and to present the fabric of SA in S-MCE, we suggest a 3-dimensional representation of SA, called the SA cube (Figure 1).

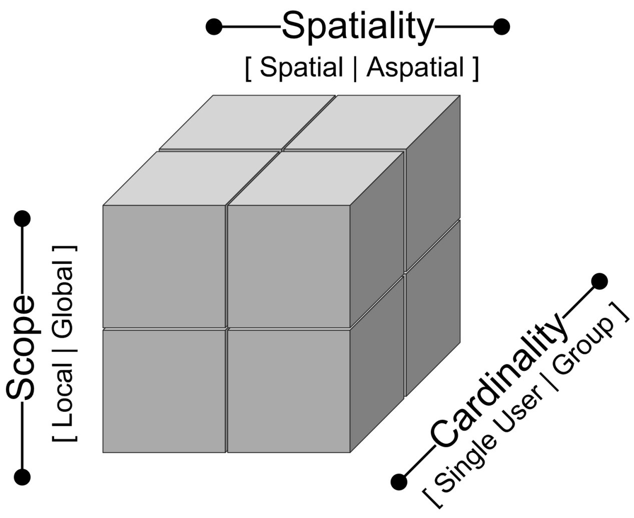


Fig. 1. Sensitivity analysis cube of spatial multiple criteria evaluation

2.1 The SA Cube

Each of the axes in Figure 1 represents one of the following characteristics of SA in S-MCE: spatiality, scope, and cardinality. *Spatiality* stresses the importance of spatial criteria and spatial weights that should be explicitly considered in GIS-based MCE. *Scope* relates to the extent of SA ranging from local one-at-a-time criterion examination to global testing of interdependencies among the multiple criteria model components. *Cardinality* addresses the duality of motivation for performing SA, namely, single-user learning and group consensus building.

Spatiality. The spatiality axis comprises both the aspatial and spatial nature of the decision situation and falls under the rubric of technicality proposed by Belton and Stewart [6]. Based on the inventory by Delgado and Sendra [14], the vast majority of the reported SA studies concern only the aspatial nature of the decision situation. Within this category of SA methods, the major factors analyzed relate to: the diversity of choice alternatives (option list), the choice of attributes (criteria), the stability of solution to changes in weights (weighting), and the uncertainty of evaluation method (e.g. standardization, weighting, and aggregation techniques). The majority of evaluation methods come from general decision theory and embrace spatial variability only implicitly. For instance, in the well-established GIS procedure of weighted overlay, the decision maker *de facto* performs traditional weighting by assigning the same importance value to every spatial unit of a given criterion layer [36,38].

Only recently it has been recognized that a spatially explicit decision component may potentially influence the rank-order of alternatives [18,25,45]. Spatial SA involves the use of topological and non-topological relations in S-MCE (Figure 2). For example, the analyst may use GIS to calculate distance between spatial decision alternatives and some attractor. The distance criterion can be further analyzed using traditional SA, for example, by changing its importance and recalculating the weighted option utility. However, given the spatial nature of the problem, the analyst should seek a more geographically oriented SA. Continuing the example, he/she may perform SA by varying the criterion importance over space and assigning different weights to different locations [18]. We call this *spatial weighting* (spatial bias), which assigns non-uniform weights to spatial units [25], providing a way of articulating stakeholders' sense of place. Each decision participant may have an individual spatial frame of reference like home, work, daily activity route, or other place of importance, which impacts the perception of the proposed courses of action. Note that this concept of spatial weight involves the perception (judgment) of criterion importance varying over geographical space and it is different from another concept of spatial weights that measure the level of interaction between features in geographical space [21].

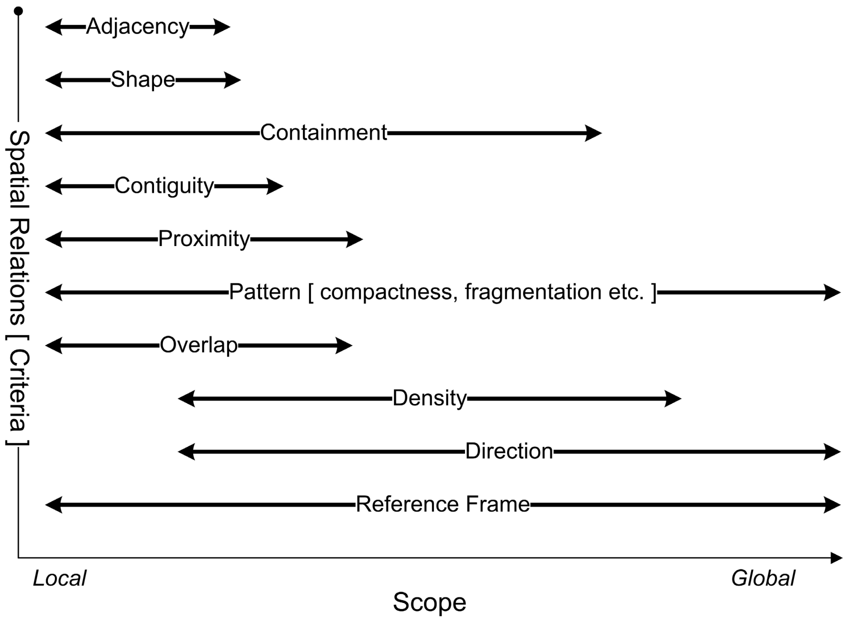


Fig. 2. Spatial relations applicable to SA

A different type of problem arises from the spatial distribution of options and their criteria values. Following the above example, the analyst may be more interested in options that form a cluster because their proximity may reinforce positive or negative effects. Such spatial characteristics (contiguity, compactness, proximity etc.) are called *spatial criteria* [8,40,45] and are derived from various spatial relations (Figure 2). Consequently, SA applied to spatial criteria does not involve subjective perception of place (spatial weighing), but rather the uncertainty of spatial distributions, interactions, impacts, and relationships, which can be studied by performing repetitive spatial transformations.

Unlike the geographical SA defined by Lodwick et al. [36], we argue that spatially explicit SA pertains rather to relative location than to spatial coincidence of absolute location. Absolute location manifests itself in the traditional SA approach of weighted overlay, where the composite score of geographic option is derived by weighting and re-weighting criteria values *at* specific locations. Additionally, it involves the uncertainty of criteria evaluation scores measured at particular sites. Relative location refers to geographical variability, which manifests itself via situational relations (e.g. contiguity, compactness, proximity) representing spatial organization and spatial configuration *in reference to* a particular location [11]. Within the context of GIS, such spatially explicit SA complements traditional (aspatial) SA.

Scope. Spatiality concerns the structure of the decision process and, in particular, the spatial component of this structure. However, if the objective is to understand the behavior of a selected subset of the criteria and their weights in the decision model (one, few, many, or all) then the scope of SA is more relevant.

Scope may range from a detailed component-focused study to a generalized simultaneous testing of interdependencies among decision elements. Conventionally, the practice of S-MCE treats SA as a method of examining one specific component of S-MCE, where the analyst changes one parameter at a time and evaluates how sensitive the output is to the change [37,39]. Additionally, the analyst may vary multiple factors but within a small range around the favored values [47]. This type of SA is very interactive in nature and is termed local SA [47]. Conversely, if we perturbate one factor within its whole distribution or vary multiple factors simultaneously over the entire problem space, we perform global SA [49]. The latter is much more data intensive and therefore is usually done through a sampling-based simulation [47].

The above local-global categorization of SA is specific to an aspatial analysis. The spatiality of geographic problems, however, puts the scope of SA in a different context. The division into local spatial SA and global spatial SA is similar to the local-global notion in spatial statistics. Specifically, spatial SA is defined as the examination of one or more spatial relations (Figure 2) within the extent of either the proximal (neighborhood) space or the whole space of the study area [36]. For example, adjacency and proximity are more neighborhood-related and should be used when dealing with local spatial associations. Conversely, the reference frame may span from a single location to the whole study area. The latter results in a constant spatial weight value, which is simply the traditional weighting in weighted overlay.

Cardinality. The third dimension of the cube, cardinality, reflects two potential motivations for SA, namely, 1) insight into individual's values, and 2) learning about the group values. Cardinality follows the line of thought proposed by Belton and Stewart [6], who divided SA into two distinct perspectives, namely single user and group. The single user perspective of SA deals with intuition and understanding. In this respect, we study the convergence or divergence of our favored options with the options suggested by the model [6]. The user discovers his/her individual viewpoint of the decision problem. In the context of S-MCE, this may manifest itself in the individualistic spatial weighting. Group SA follows a different logic and is concerned more with the perspectives of others than with the individual perspective. Whereas single-user SA has its roots in operations research, the group SA is more related to collaborative learning. Given the value-laden nature of group decision making, group SA should incorporate analytic-deliberative functionality [43], to enhance place-based qualitative perceptions and consensus building among stakeholders [2,19,26]. A possible quantitative application of group SA in S-MCE relates to the analysis of spatial equity [52].

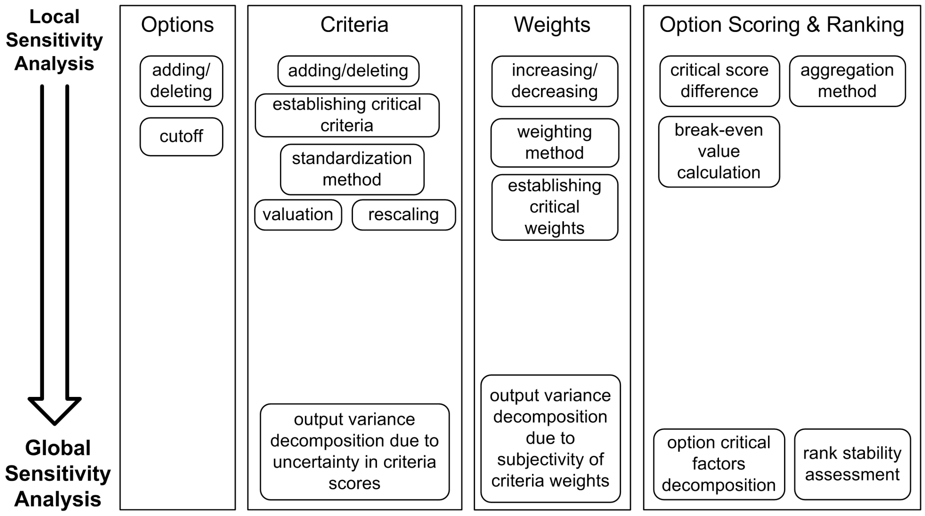
3 Methods and Measures of SA

This section examines various methodological approaches to SA within the SA cube framework (Figure 1). Since none of the methods developed so far is all-inclusive, and since different methods of SA produce different outcomes, a good understanding of SA methodology and trade-offs involved in using different methods is needed in order to effectively apply SA [1,3,47].

3.1 Traditional Local and Global SA Methods

Given an initial decision option prioritization, we may focus on analyzing the sensitivity of various components that contribute to computing the rank-order. Therefore, we divided the methods into four broad, partly overlapping categories of options, criteria, weights, and option scoring/ranking (Figure 3).

The first group of methods concentrates on modifying the list of examined options. The analyst may delete some options which score low or are otherwise



Local Sensitivity Analysis
Global Sensitivity Analysis
Options
adding/ deleting
cutoff
Criteria
adding/deleting
establishing critical criteria
standardization method
valuation
rescaling
output variance decomposition due to uncertainty in criteria scores
Weights
increasing/ decreasing
weighting method
establishing critical weights
output variance decomposition due to subjectivity of criteria weights
Option Scoring & Ranking
critical score difference
aggregation method
break-even value calculation
option critical factors decomposition
rank stability assessment

Fig. 3. Aspatial SA methods and their scope

inferior or he/she may add back the previously deleted option. It is also possible to impose a constraint on a criterion value and thus modify the ranking by removing these options that do not meet the constraint [2].

Similarly to adding/deleting options, we may modify the rank-order by deleting or adding a criterion. According to Alexander [1] a criterion that changes the order of the best option (by being added or deleted) may be termed sensitive. We modify the list of criteria if they do not reflect our values and preferences [57]. In weighted overlay, changing the attribute list is equivalent to map removal sensitivity [36]. Additionally, we might be uncertain about criteria evaluation scores [57]. This problem is best addressed by using ranges/distribution of values rather than a specific value. With such constructed parameters, we may perform a Monte Carlo (MC) simulation of output variability due to the uncertainty of criteria scores [46].

Criterion uncertainty may also be related to a standardization method used to convert criteria to a common scale [1,46] or the valuation of a criterion [31]. For example, a person may choose to maximize the proximity to a proposed transportation improvement project since the project will benefit him/her by shortening his/her daily commute. Another person may deem such a proximity criterion as cost because it reduces public safety and hence, he/she wants to locate the projects elsewhere (minimize proximity). A novel method of determining the most critical decision criteria was proposed by Triantaphyllou [55], where the criticality is defined by the minimum change in performance measures (evaluation scores) causing the rank reversal for any two options. For the highly uncertain criteria it may also be useful to perform rescaling [57] from a higher to a lower measurement level, for example from ratio/interval to ordinal scale. The resultant decrease in accuracy may in fact better reflect the true reliability of data.

Weights have often been criticized as the subjective component of S-MCE and hence have been the focus of SA methods. The basic method of examining the sensitivity of weights relates systematic changes in weight values to changes in option ranking. Weighting plays a double role in S-MCE - it either represents a relative criterion importance or a substitution rate among criteria [18]. Due to the large number of explicit weighting methods that have been proposed [39], the choice of weighting method can also be a subject of SA [46]. It is also possible to determine critical weights for these criteria for which a relatively small change in the weight value causes the rank reversal of any two options [7,55]. Additionally, for one parameter at a time weight change analysis, there is a rule of thumb for establishing criterion criticality, which states that if the decision maker changes the parameter by n-percent and the result will change by less than n-percent then he/she may conclude that this parameter does not significantly influence the result [37].

The methods so far discussed are mainly local in scope. A more complex approach to establishing the importance of criteria scores and weights involves global SA, which decomposes the variance of the output of the MCE process into a variety of explanatory factors. An example of a global SA method is the extended Fourier Amplitude Sensitivity Test (FAST). The extended FAST method uses *first* and *total* order indices as SA measures. The first order sensitivity index is defined as a fractional contribution to the variance of MCE model output (e.g. option ranking) due to the uncertainty of a given input parameter treated independently from other parameters [13,22,46,49,50]. The total order index represents the overall contribution of a given parameter (e.g. criterion weight) including its interactions with other parameters. Computation of the indices requires a large number of rank-order calculations performed with weight vectors derived from the decision maker's weight distribution functions [46].

The key challenge for SA in S-MCE is to determine the stability of the rank-order of decision options. While all four groups of aspatial SA methods in Figure 3 can be used to determine the stability of rank-order, the option scoring and ranking methods address this task directly. Combining weighted multiple criteria values is inherently uncertain since none of the developed aggregation methods is flawless. Thus, SA may also comprise of comparing the stability of the rank-order under different aggregation methods [33,41].

Sometimes, highly ranked decision options are very close to each other in terms of their overall evaluation scores being very similar. Such a situation warrants a careful investigation of critical score difference (Figure 3). For these options, it may be interesting to discover what changes in weights make them score equally well, a procedure which Pannell [44] calls the *break even value*. If the break even value is within an acceptable range, then we may justify the switch in rank and select an option which we value more given other intangible criteria. The scale of the necessary weight modification may also provide information about options that are more likely to be ranked first - a method called proximity ranking [58].

Insua and French [27] proposed a more generic framework for SA scoring and ranking. According to their framework, we should first find the non-dominated alternatives, then narrow down the non-dominated set to those alternatives that are optimal, find the adjacent potentially optimal alternatives, and establish for them the least change in weights that is needed to switch the highest ranked option. Such analysis not only pinpoints the most efficient options and their competitors, but also provides information about the minimum tradeoffs between the options [19]. A more extensive SA of option scoring relies on the division of multidimensional space into subsections of weight value ranges where a selected favored alternative wins [6]. These subsections are then called preference regions. Sampling-based MC simulation is another useful approach to assessing rank stability [20,37,39,57]. For example, for a large number of rank-orders from MC runs we can calculate different summary statistics like minimum, maximum, and mean position of the option thus revealing how stable the option rank is under changing parameter values [3].

3.2 Spatial SA Methods

Spatial SA is an underdeveloped component of SA in S-MCE. Studies that utilize spatially-explicit SA are rare. Table 1 is a proposition of a GIS-based methodology that addresses spatially-explicit components of S-MCE problems. Based on these components, spatial SA methods are grouped into options, criteria, and weights-focused methods.

Table 1. Methods supporting spatial SA

S-MCE Component Sensitivity	Spatial Relations	SA Procedure	Example Operations
OPTIONS: Spatial distribution of options e.g. dispersed, contiguous	Adjacency	Map options that are adjacent	Topology operations e.g. 'Share a line segment'
	Proximity	Locate proximate options	Spatial statistics e.g. measures of centrality, cluster analysis; Distance decay functions
	Pattern	Map the dispersion of options and the shape of option clusters	Spatial statistics: mapping clusters; Point/line/kernel density; Map algebra: neighborhood operations
	Direction	Map the direction options distribution	Directional distrbution; Directional mean
CRITERIA: spatial distribution of evaluation scores	Pattern	Modify the criterion layer with a random uncertanty layer derived from spatial distribution of this criterion	Spatial statistics: cluster analysis, spatial autocorelation; Point/line/kernel density; Spatial interpolation
		Use a more generalizedlized criterion layer	Map algebra: neighborhood operations; Reclassify/remap; Aggregate
WEIGHTS: Reference frame (point, line, or area of interest) and its spatial distribution	Proximity	Use different reference frames as spatial weight layers	Straight line distance; Distance decay functions; Spatial statistics: standard distance, standard deviational ellipse
	Containment		Variable-size buffer; Overlay: e.g. point-on-polygon; Map algebra: Boolean and zonal operations

The first group of spatial SA methods refers to geographical distributions of the decision *options* represented in Table 1 by four categories of relations: adjacency, proximity, pattern, and direction. For example, the decision maker may prefer to select high-ranked options because they are located in the direct proximity to other high-ranked options [45]. Such spatial autocorrelation can lead to positive spatial externalities. Alternatively, the decision maker may use spatial statistics to analyze and map significant clusters of high scoring options. Other spatial SA operations, pertaining to spatial distribution of options, include map algebra, topology rules, or density analysis. Such operations can be used to derive geographically adjusted option evaluation scores [45].

The sensitivity of decision *criteria* stems from the uncertainty of the evaluation scores (criteria values). In spatial decision situations, such sensitivity could be analyzed based on traditional value measurement uncertainty (error) or value assessment ambiguity, and spatial uncertainty stemming from the geographical distribution of a given criterion. For example, if the decision maker uses rainfall as a decision criterion in raster-based S-MCE, he/she should consider the uncertainty of the rainfall measure at a particular location, together with the uncertainty associated with a selected interpolation method. One way of analyzing the spatial sensitivity of a criterion involves adding an uncertainty surface to the criterion surface. The uncertainty surface should be derived from a spatial distribution of the criterion under consideration [32]. Another method of analyzing criterion spatial sensitivity involves attribute generalization [32], similar to the more traditional rescaling proposed by Voogd [57].

Spatial *weight* sensitivity pertains to the subjective reference frame of the decision maker. Participants may perceive certain locations as more favorable from a given perspective. For example, they may conceive the rainfall criterion as being more important in rural areas than elsewhere. Consequently, spatial sensitivity of criteria weights may have a direct effect on option scoring and ranking and should be explicitly addressed. Two spatial relations that capture the spatial sensitivity of criteria weights are *the proximity of* and *the containment within* the areas of interest (Table 1). The former can be applied using different distance-decay functions [45] or modeling situation factors [12]. The latter can be implemented by a variable-size buffer analysis [23,32,53].

3.3 Methods Supporting Group SA

The SA methods discussed above support the dynamics of group SA only implicitly. When used in a collaborative setting, the results of individual SA must be aggregated by some means. Jankowski et al. [31] and Feick and Hall [18] present consensus maps that show the dispersion of summarized votes using graduated circles or colors. The votes relate to a number of decision elements like option scores, criteria weights or criteria selection. Borda or Copeland voting protocols are used to aggregate the votes. Additionally, the variance of voting results can be displayed to identify the most contentious issues that need further discussion. These analytical techniques rely solely on non-spatial social voting functions. Spatially-explicit and scope-variant group SA remains to be explored in future research.

4 Discussion and Challenges

Like any method of S-MCE, SA should embody a range of techniques that fit multiple styles of decision making [2,20,42] and, thus, the need for appropriate SA methods is self-evident. Many well-developed techniques for SA in S-MCE exist; however, each of them has limitations.

The aspatial methods of SA have been used for years and there is an extensive record of SA applications. However, these methods ignore a geographic aspect of S-MCE expressed by the spatial variability of criteria values and weights. We have suggested that spatial variability and hence, the sensitivity of S-MCE solutions to criteria values and weights can be addressed by the analysis of spatial relations. The methodology of spatial SA is currently still in its infancy and lacks techniques for rank stability testing. Furthermore, not much is known on whether spatial SA can enhance group decision making [18].

Until recently, the majority of studies reported on local one parameter at a time SA techniques. Local SA proved to be useful for human-mediated dynamic testing of rank-order stability. Yet, local SA makes sense only if we deal with perfectly linear models [50], which is unlikely in the majority of spatial decision problems. The most appropriate technique for non-linear processes is the model-independent variance-based global SA [46]. Also, due to its theoretical underpinnings, global SA constitutes an important advance over the local SA methods.

It is also worth noting the role of qualitative group SA, in which the stakeholders negotiate using in-depth descriptive information that may reinforce or change their initial selection. Although only mentioned in this paper, such *soft* SA presents an intriguing area for future research and has the potential to bridge quantitative and qualitative approaches in decision making [34].

4.1 Visual Representations of SA

SA is a useful approach to explore spatial decision options, but its utility can be further enhanced by the use of a cognitively straightforward visual feedback to decision makers. In this respect, the greatest progress has been made in the techniques of traditional local SA. The developments have been based on the principles of human-computer interaction and include user-friendly interfaces, computationally efficient algorithms, data brushing and dynamic linking between the maps, tables, and graphs. A variety of tools are available in today's software including weight sliders [3,29], value path plots [2,6,44], spider-web charts [44], appraisal roses [57], or pie and bar chart histograms [10]. More complex display methods have been proposed to account for the interrelated multidimensional nature of a decision problem. Examples include graphs of time-variant decomposition [50], decision maps [30], and policy landscapes combined with policy regions [5].

A future research challenge for visual representations concerns effective techniques of representing the sensitivity of S-MCE solutions at spatial locations. The exemplary cartographic representations of SA in the form of rank maps, rank stability maps, or utility symbol maps should be enhanced with more specific sensitivity maps.

4.2 Representational and Computational Issues

Krivoruchko and Gotway-Crawford [32] notice that the results of uncertainty and SA may be numerous and call for creative summary tools. Standards for defining statistical descriptors and improvements in computational techniques bode well for enhancing the ease of use and comprehensiveness of SA [18]. An exemplary possibility is the dynamic visualization of global SA in the form of a pre-computed approximation of a solution hypercube, which is dynamically sliceable depending on the specified parameter set. SA creates new information about the decision process [44] and, within GIS, this emergent information can be introduced in the form of maps [30]. Another important issue to be considered relates to the spatial and aspatial scales of data granularity, representative to a specific decision situation. A possible solution may involve a nondeterministic fuzzy approach to SA in S-MCE [17].

5 Conclusions

This article began with declaring SA as the simultaneous advantage and shortcoming of S-MCE. In the course of the article, we presented a framework for a holistic SA within the context of S-MCE. The goal of this framework is to assist the decision makers with the selection of SA methods and techniques that are appropriate for a specific decision situation. We formalized the structure of SA into spatiality, scope, and cardinality. The first dimension recognizes the duality of the spatial and aspatial nature of geographic decision making. The second dimension relates to analytical methods ranging from the low level exploration of a single decision parameter to the global synthesis of decision situation sensitivities. The final dimension reflects the divergent motivation for performing SA, namely individual learning or group consensus building.

We also discussed a number of SA techniques, delineating the advantages and disadvantages associated with parametric and nonparametric (judgmental) uncertainty. While the spatiality of alternatives is widely recognized, little progress has been made on including spatial aspects of decision criteria and decision weights into MCE-based SA. Therefore, the development of spatially explicit techniques of SA deserves further research. Additionally, the SA cube framework points to the lack of both spatial and aspatial scope-variant group SA methods, which are especially needed to enhance collaborative decision making.

SA is not a substitute for decision making analysis. Instead, it is a way of making the participants of the decision processes aware of the uncertainties inherent in any decision situation. Unlike a *typical* perception of scientific analysis, in which we put probably too much trust in precise (but not necessarily accurate) results, SA emphasizes the impossibility of providing an *always-best* solution [48]. Ascough et al. [4] state that multicriteria SDSS should be an environment where decision makers can define, explore, redefine and understand the problems they deal with. To accomplish this goal, we need more research into the methods of spatial SA. The SA framework presented here is intended to be a step in building the foundation for further progress in spatial decision support.

Acknowledgments. This material is based on research supported in part by National Science Foundation Grant No. EIA 0325916, funded through the Information Technology Research Program, and managed by the Digital Government Program. Any opinions, findings, and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The support of NSF is gratefully acknowledged. The authors would also like to thank three anonymous reviewers who provided valuable comments on an earlier version of this manuscript.

References

1. Alexander, E.R.: Sensitivity Analysis in Complex Decision Models. Journal of American Planning Association 55, 323–333 (1989)
2. Andrienko, G., Andrienko, N., Jankowski, P.: Building Spatial Decision Support Tools for Individuals and Groups. Journal of Decision Systems 12, 193–208 (2003)
3. Andrienko, N., Andrienko, G.: Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach. Springer, Berlin (2005)
4. Ascough II, J.C., Rector, H.D., Hoagh, D.L., McMaster, D.L., Vandenberg, B.C., Shaffer, M.J., Weltz, M.A., Ahja, L.R.: Multicriteria Spatial Decision Support Systems: Overview, Applications, and Future Research Directions. In: Rizzoli, A.E., Jakeman, A.J. (eds.) Integrated Assessment and Decision Support, Proceedings of the First Biennial Meeting of the iEMSs, vol. 3, pp. 175–180 (2002)
5. Bankes, S.C.: Tools and techniques for developing policies for complex and uncertain systems. In: PNAS 1999, pp. 7263–7266 (2002)
6. Belton, V., Stewart, T.J.: Multiple Criteria Decision Analysis An Integrated Approach. Kluwer Academic Publishers, Dordrecht (2002)
7. Bojórquez-Tapia, L.A., Sánchez-Colon, S., Martinez, A.F.: Building Consensus in Environmental Impact Assessment Through Multicriteria Modeling and Sensitivity Analysis. Environmental Management 36, 469–481 (2005)
8. Brookes, C.J.: A parameterized region growing program for the site allocation on raster suitability maps. International Journal of Geographical Information Science 11, 375–396 (1997)
9. Carver, S.J.: Integrating multi-criteria evaluation with geographical information systems. International Journal of Geographical Information Systems 5, 321–339 (1991)
10. Chen, K., Blong, R., Jacobson, C.: MCE-RISK: integrating multicriteria evaluation and GIS for risk decision-making in natural hazards. Environmental Modelling and Software 16, 387–397 (2001)
11. Couclelis, H.: Requirements for planning-relevant GIS: a spatial perspective. Papers in Regional Science The Journal of the RSAI 70, 9–19 (1991)
12. Cromley, R.G., Huffman, F.T.: Modeling Situation Factors Used in MCE Procedures for Raster GIS. Transactions in GIS 10, 239–251 (2006)
13. Crosetto, M., Tarantola, S.: Uncertainty and sensitivity analysis: tools for GIS-based model implementation. International Journal of Geographical Information Science 15, 415–437 (2001)
14. Delgado, M., Sendra, J.B.: Sensitivity Analysis in Multicriteria Spatial Decision-Making: A Review. Human and Ecological Risk Assessment 10, 1173–1187 (2004)

15. Densham, P.J.: Spatial Decision Support Systems. In: Longley, P.A., Goodchild, M.F., Maguire, D.J., Rhind, D.W. (eds.) Geographical Information Systems: Principles and Applications, 1st edn., pp. 403–412. Longman, London (1991)
16. DiBiase, D., Demers, M., Johnson, A., Kemp, K., Taylor-Luck, A., Plewe, B., Wentz, E.: Geographic Information Science and Technology Body of Knowledge. UCGIS, AAG (2006)
17. Eastman, J.R.: Multi-criteria evaluation and GIS. In: Longley, P.A., Goodchild, M.F., Maguire, D.J., Rhind, D.W. (eds.) Geographical Information Systems: Principles, Techniques, Management and Applications, 2nd edn., pp. 493–502. Wiley, New York (1999)
18. Feick, R.D., Hall, G.B.: A method for examining the spatial dimension of multi-criteria weight sensitivity. International Journal of Geographical Information Science 18, 815–840 (2004)
19. French, S.: The role of sensitivity analysis in decision analysis. In: Executive Information Systems and Decision Support, UNICOM, Applied Information Technology, vol. 15, pp. 99–123. Chapman and Hall, Boca Raton (1992)
20. French, S., Xu, D.-L.: of Multi-attribute Decision Analytic Software. Journal of Multi-Criteria Decision Analysis 14, 65–80 (2005)
21. Getis, A., Aldstadt, J.: Constructing the spatial weights matrix using a local statistic. Geographical Analysis 36, 90–104 (2004)
22. Gómez-Delgado, M., Tarantola, S.: GLOBAL sensitivity analysis, GIS and multi-criteria evaluation for a sustainable planning of a hazardous waste disposal site in Spain. International Journal of Geographical Information Science 20, 449–466 (2006)
23. Halls, J.: A Spatial Sensitivity Analysis of Land Use Characteristics and Phosphorus Levels in Small Tidal Creek Estuaries of North Carolina, USA. Journal of Coastal Research 36, 340–351 (2002)
24. Helton, J., Burmaster, D.E.: Guest Editorial: treatment of aleatory and epistemic uncertainty in performance assessments for complex systems. Reliability Engineering and System Safety 54, 91–94 (1996)
25. Herwijnen, M.V., Rietveld, P.: Spatial Dimensions in Multicriteria Analysis. In: Thill, J.-C. (ed.) Spatial Multicriteria Decision Making and Analysis A geographic information sciences approach, pp. 77–99. Ashgate, London (1999)
26. Insua, D.R.: Introduction to Special Issue on Sensitivity Analysis. Journal of Multi-Criteria Decision Analysis 8, 117–118 (1999)
27. Insua, D.R., French, S.: A framework for sensitivity analysis in discrete multi-objective decision-making. European Journal of Operational Research 54, 179–190 (1991)
28. Jankowski, P.: Integrating Geographical Information Systems and Multiple Criteria Decision-Making Methods. International Journal of Geographical Information Systems 9, 251–273 (1995)
29. Jankowski, P., Andrienko, N., Andrienko, G.: Map-centered exploratory approach to multiple criteria spatial decision making. International Journal of Geographical Information Science 15, 101–127 (2001)
30. Jankowski, P., Lotov, A., Gusev, D.: Application of Multiple Criteria Tradeoff Approach to Spatial Decision Making. In: Thill, J.-C. (ed.) Spatial Multicriteria Decision Making and Analysis A geographic information sciences approach, pp. 127–148. Ashgate, London (1999)
31. Jankowski, P., Nyerges, T.L., Smith, A., Moore, T.J., Horvath, E.: Spatial group choice: a SDSS tool for collaborative spatial decision-making. International Journal of Geographical Information Science 11, 577–602 (1997)

32. Krivoruchko, K., Gotway-Crawford, C.A.: Assessing the Uncertainty Resulting from Geoprocessing Operations. In: Maguire, D., Batty, M., Goodchild, M. (eds.) GIS, Spatial Analysis, and Modeling, pp. 67–92. ESRI Press, Redlands (2005)
33. Laaribi, A., Chevallier, J.J., Martel, J.M.: A spatial decision aid: a multicriterion evaluation approach. Computers, Environment, and Urban Systems 20, 351–366 (1996)
34. Lempert, R.J.: A new decision science for complex systems. In: PNAS, vol. 99, pp. 7309–7313 (2002)
35. Lempert, R.J., Pooper, S.W., Bankes, S.C.: Shaping the Next One Hundred Years New Methods for Quantitative, Long-Term Policy Analysis, The RAND Pardee Center, Santa Monica (2003)
36. Lodwick, W.A., Monson, W., Svoboda, L.: Attribute error and sensitivity analysis of map operations in geographical information systems: suitability analysis. International Journal of Geographical Information Systems 4, 413–428 (1990)
37. Longley, P.A., Goodchild, M.F., Maguire, D.J., Rhind, D.W.: Geographical Information Systems: Principles, Techniques, Management and Applications, 2nd edn. Wiley, New York (2005)
38. Lowry, J.H., Miller, H.J., Hepner, G.F.: A GIS-Based Sensitivity Analysis of Community Vulnerability to Hazardous Contaminants on the Mexico/U.S. Border. Photogrammetric Engineering and Remote Sensing 61, 1347–1359 (1995)
39. Malczewski, J.: GIS and Multicriteria Decision Analysis. Wiley, New York (1999)
40. Malczewski, J.: GIS-based multicriteria decision analysis: a survey of the literature. International Journal of Geographical Information Science 20, 703–726 (2006)
41. Massam, B.: The location of waste transfer stations in Ashdod, Israel, using a multi-criteria Decision Support System. Geoforum 22, 27–37 (1991)
42. Merkhofer, M.W.: Assessment, Refinement, and Narrowing of Options. In: Dale, V.H., English, M.R. (eds.) Tools to Aid Environmental Decision Making, pp. 231–281. Springer, New York (1998)
43. Nyerges, T.L., Ramsey, K.S., Wilson, M.W.: Design Considerations for an Internet Portal to Support Public Participation in Transportation Improvement Decision Making. In: Dragicevic, S., Balram, S. (eds.) Collaborative GIS, pp. 208–236. Idea Group Inc., Hershey (2006)
44. Pannell, D.J.: Sensitivity Analysis of normative economic models: theoretical framework and practical strategies. Agricultural Economics 16, 139–152 (1997)
45. Rinner, C., Heppleston, A.: The Spatial Dimensions of Multi-Criteria Evaluation - Case Study of a Home Buyer's Spatial Decision Support System. In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 338–352. Springer, Heidelberg (2006)
46. Saisana, M., Saltelli, A., Tarantola, S.: Uncertainty and sensitivity analysis techniques as tools for the quality assessment of composite indicators. Journal of Royal Statistical Society A 168, 307–323 (2005)
47. Saltelli, A., Chan, K., Scott, E.M.: Sensitivity Analysis. Wiley, New York (2000)
48. Saltelli, A., Tarantola, S., Campolongo, F., Ratto, M.: Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models. Wiley, New York (2004)
49. Saltelli, A., Tarantola, S., Chan, K.: A Quantitative Model-Independent Method for Global Sensitivity Analysis of Model Output. Technometrics 41, 39–56 (1999a)
50. Saltelli, A., Tarantola, S., Chan, K.: A Role for Sensitivity Analysis in Presenting the Results from MCDA Studies to Decision Makers. Journal of Multi-Criteria Decision Analysis 8, 139–145 (1999b)
51. Simon, H.: The new science of management decision. Harper, New York (1960)

52. Talen, E.: Visualizing fairness: Equity maps for planners. Journal of American Planning Association 64, 64–75 (1998)
53. Tarantola, S., Giglioli, N., Jesinghaus, J., Saltelli, A.: Can global sensitivity analysis steer the implementation of models for environmental assessments and decision-making? Stochastic Environmental Research and Risk Assessment 16, 63–76 (2002)
54. Thill, J.C.: GIS and Multiple Criteria Decision Making: A Geographic Information Science Perspective, Ashgate, London (1999)
55. Triantaphyllou, E.: Multi-Criteria Decision Making Methods: A Comparative Study. Kluwer Academic Publishers, Dordrecht (2000)
56. Uran, O., Janssen, R.: Why are spatial decision support systems not used? Some experiences from the Netherlands. Computers, Environment, and Urban Systems 27, 511–526 (2003)
57. Voogd, H.: Multicriteria Evaluation for Urban and Regional Planning. Pion Limited, London (1983)
58. Wolters, W.T.M., Mareschal, B.: Novel types of sensitivity analysis for additive MCDM methods. European Journal of Operational Research 81, 281–290 (1995)

Reasoning on Spatial Relations between Entity Classes

Stephan Mäs

AGIS - Arbeitsgemeinschaft GIS, University of the Bundeswehr Munich,
Werner Heisenberg Weg 39, 85577 Neubiberg, Germany
{Stephan.Maes}@unibw.de

Abstract. The facilitation of interoperability requires a clear distinction if a relation refers to classes of individuals or to specific instances, in particular when it comes to the logical properties of the involved relations. Class relations are defined whenever the semantics of entire classes are described, independently of single instances. Typical examples are spatial semantic integrity constraints or ontologies of entity classes. The paper continues research on spatial class relations by deepening the analysis of the reasoning properties of class relations. The work is based on a set of 17 abstract class relations defined in [11]. The paper provides a complete composition table for the 17 abstract class relations and redefines the concept of conceptual neighbourhood for class relations. This approach can be used to find conflicts and redundancies in sets of semantic integrity constraints or other applications of spatial class relations.

Keywords: Class Level Relations, Spatial Relations, Reasoning, Compositions, Conceptual Neighbourhood, Constraint Networks.

1 Introduction

The definition of class relations and their logical properties did not attract much attention of the scientific community so far. But as already argued by [1] the facilitation of interoperability requires a clear distinction if a relation refers to classes of individuals or to concrete instances, in particular when it comes to the logical properties of the involved relations. The difference becomes obvious through simple natural language statements, like for example: "house #12 is contained by parcel #1234". This is a simple statement about two entities related by the spatial relation *contained by*. Since *contained by* is the inverse relation of *contains* the statement also implies "parcel #1234 contains house #12". A statement about the corresponding entity classes is "buildings are contained by a parcel". Applying the symmetry of the instance relation again it becomes: "a parcel contains buildings". These statements can be mistaken, since they should be understood as "every building is contained by some parcel", but NOT as "every parcel contains some building". This example shows the influence of words like "all" or "some" on the semantic of a statement. They define cardinality restrictions on the applied relation. For a human reader it is very often possible to interpret the correct semantics, but a formal description of such a statement must explicitly contain cardinality information. The example also shows that a relation among the classes is not subject to the same logical properties as a relation between instances and the cardinality information must be considered for reasoning. This has also been pointed out by [1] and [11].

Class relations define a cardinality restriction for a certain relation between the individuals of classes. The restriction is always valid for entire classes or subsets of classes and not exclusive for single instances. A class relation always links the cardinality restriction to an instance relation. Typical applications of class relations are ontologies of classes and semantic integrity constraints. For geographical information class relations are of particular interest, because a semantic description of such data requires class relations which are based on spatial instance relations, like e.g. topological or metric relations. The interoperable exchange of data of different domains and application areas requires semantic descriptions of the data. Class relations are useful for the formalization of these descriptions. The logical properties of the class relations support an automatic processing, querying and comparing of such descriptions.

As shown in previous investigations [1] [11] it is useful to separately analyse the reasoning properties of the class relations and those of the instance relations. Therewith class relations can be flexibly used in combination with any kind of instance relation. In a previous paper a set of 17 abstract class relations has been defined, which is independent of concrete instance relations [11]. For these relations some basic reasoning concepts have been investigated. This paper continues these investigations particularly with regard to the composition of class relations and the conceptual neighbourhood of class relations. The following chapter will recapitulate the definition of basic cardinality properties and based on that the definition of the 17 abstract class relations. In the third chapter the composition of 17 class relations is investigated. The paper provides an overall analysis of all possible compositions, which has never been presented before. Furthermore the concept of conceptual neighbourhood, which has been so far only considered in combination with instance relations, is redefined for class relations. It is shown how these logics can be used to find conflicts in triples of class relations.

2 Definition of Class Level Relations

A class relation defines cardinality restrictions for a certain relation between all instances of the involved classes. In the following subchapters some basic cardinality properties are defined and used for the definition of class relations. It recapitulates the definitions made in [11]. For a more extensive elaboration refer to the original paper.

2.1 General Definitions and Requirements

For the definition of class relations the classes must conform to the following two preconditions. First, the involved classes must have at least one instance, i.e. empty classes are not feasible. As stated before class relations are linked to instance relations. Thus the second condition specifies that if a class relation is defined, there must at least one corresponding instance relation exist among the instances of the involved classes. The investigations in this paper are restricted to binary relations between entire entity classes. Relations between three or more classes or between subsets of classes (e.g. all blue houses as a subset of the class house) are not considered.

In the following definitions small letters ‘x’, ‘y’, ‘z’, … denote variables for instances / individuals. Every instance must be associated to an entity class. For entity classes capital letters ‘A’, ‘B’, ‘C’, … are used. ‘Inst(x,A)’ means individual x is an instance of class A. The function ‘r(x,y)’ means instance x has the relation r to instance y; x and y are said to participate in the relationship instance r. The meta-variable r can stand for any relation between instances (e.g. topological relations between areal features [2], which are used in the examples the paper). Instance relationships can be associated to a class relation R. For class relation definitions ‘R(A,B)’ denotes that R relates the classes A and B. The meta-variable R can stand for any class relationship. Every R is related to an instance relation r. If a class relation R(A,B) is defined, at least one r must exist between the instances of A and B.

2.2 Cardinality Properties of Class Level Relations

Cardinalities represent the number of elements of a set. Class relations refer to an instance relation and restrict the cardinality of this relation between the instances of the involved classes. The restrictions are defined through cardinality properties. In [11] the following cardinality properties of class relations have been used.

$$\mathrm{LT}(A,B,r) := \forall x(\mathrm{Inst}(x,A) \rightarrow \exists y(\mathrm{Inst}(y,B) \cap r(x,y))). \tag{CP1}$$

$$\mathrm{RT}(A,B,r) := \forall y(\mathrm{Inst}(y,B) \rightarrow \exists x(\mathrm{Inst}(x,A) \cap r(x,y))). \tag{CP2}$$

The cardinality properties (CP1) and (CP2) define a totality for the class A and B respectively. (CP1) holds if every instance of A has the relation r to some instance of B. In set theory such relations are called *left-total*.

(CP2) holds if for each instance of B there is some instance of A which stands in relation r to it. This means that every instance of B has the inverse relation of r to some instance of A. In this case the relation is *right-total*. The concepts of totality have also been used for the class relations defined in [1].

$$\begin{aligned}\mathrm{LD}(A,B,r) := \forall x,y,z[&\mathrm{Inst}(x,A) \cap \mathrm{Inst}(y,B) \cap \mathrm{Inst}(z,A) \cap \\ &r(x,y) \cap r(z,y) \rightarrow x = z] \cap \mathrm{Ex}(A,B,r).\end{aligned} \tag{CP3}$$

$$\begin{aligned}\mathrm{RD}(A,B,r) := \forall x,y,z[&\mathrm{Inst}(x,A) \cap \mathrm{Inst}(y,B) \cap \mathrm{Inst}(z,B) \cap \\ &r(x,y) \cap r(x,z) \rightarrow y = z] \cap \mathrm{Ex}(A,B,r).\end{aligned} \tag{CP4}$$

$$\mathrm{Ex}(A,B,r) := \exists x \exists y(\mathrm{Inst}(x,A) \cap \mathrm{Inst}(y,B) \cap r(x,y)). \tag{CP5}$$

Class relations which hold (CP3) are *left-definite* and specify that for no instance of B there is more than one instance of A which stands in relation r to it. This property restricts the number of r relations an instance of B can participate; the instances of A are not restricted. The last term ensures that at least one instance relation r does exist between the instances of A and B (CP5).

(CP4) specifies that no instance of A participates in a relationship r to more than one instance of B. When this cardinality property is defined in a class relation all instances of A are restricted while the instances of B are not affected. The corresponding class relations are *right-definite*.

Such properties of class relations are well established in data modelling, for example when total participation and cardinality ratio constraints are described using the Entity-Relationship notation. In such models a total participation is represented by a double line for the relation and cardinality ratio for example by a N:1 next to the relation signature (figure 1). In this example all buildings are restricted to be contained by exactly one parcel, while the parcels are allowed to contain an undefined number of buildings. *Contains* is the restricted instance relation. The number of different cardinality ratio constraints of such a notation is indefinite. This approach only considers a cardinality ratio of 0..1, which is representing the concept of unambiguousness.

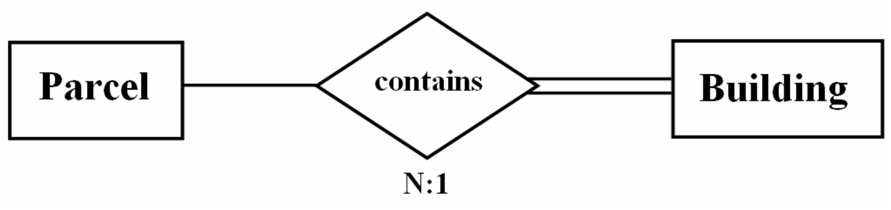


Fig. 1. Constraints in an Entity-Relationship Notation

2.3 Class Level Relations

The formal definition of class relations is based on the above defined cardinality properties *left-definite*, *right-definite*, *left-total* and *right-total.* These properties are independent of each other. This means that no property implies or precludes one of the other properties. If a class relation is only defined as *right-total* there is no information about its left totality and the unambiguousness available.

As the example in figure 1 illustrates, the properties can be combined for the definition of a class relation. The class relation in the example is based on the topological instance relation *contains* and the cardinality properties *left-definite* and *right-total.* The other two properties are not valid. The corresponding class relation *CONTAINS$_{LD.RT}$(Parcel, Building)* is based on the class relation defined in (CR1).

$$R_{\mathrm{LD.RT}}(A,B) := \mathrm{LD}(A,B,r) \cap \mathrm{RT}(A,B,r) \cap \neg \mathrm{RD}(A,B,r) \cap \neg \mathrm{LT}(A,B,r). \qquad \text{(CR1)}$$

In the following I will refer such class relations, which are not linked to a particular instance relation, as **abstract class relations** (e.g. $R_{LD.RT}$(A,B)). In analogy to (CR1) the four cardinality properties can be used to define a set of 15 abstract class relations, where for each relation at least one of the four properties holds and the others are excluded, respectively. An investigation of all possible combinations leads to:

– Four abstract class relations where one property is valid and the corresponding other three are excluded,
– Six abstract class relations where two properties are valid and the other two are excluded,
– Four abstract class relations which combine three of the four defined cardinality properties respectively and exclude the corresponding fourth,
– And one abstract class relations where all four properties are valid.

Additionally to these 15 abstract class relations two special cases are considered in [11]. To achieve a jointly exhaustive set of relations one abstract class relation is defined for the situation that none of the four properties is valid but some instances of A stand in relation r to some instances of B (CR2).

Second, a further abstract class relation is defined for the case that all instances of A have a relationship instance of R to all instances of B (CR3). This is a strict case of a *left-total* and *right-total* relation. For class relations it is frequently occurring, for example when no instances of two classes are allowed to intersect: $\text{DISJOINT}_{\text{LT.RT-all}}(\text{Streets, Lakes})$.

$$R_{\text{some}}(A,B) := Ex(A,B,r) \cap \neg LD(A,B,r) \cap \neg RD(A,B,r) \cap \neg LT(A,B,r) \cap \neg RT(A,B,r). \tag{CR2}$$

$$R_{\text{LT.RT-all}}(A,B) := \forall x \forall y (Inst(x,A) \cap Inst(y,B) \rightarrow r(x,y)). \tag{CR3}$$

This set of 17 abstract class relations enables the definition of class relations based on any binary instance relation. For further details on the definition of the abstract class relations it is referred to [11]. Figure 2 shows an example for each of the abstract class relations.

Table 1. Restrictions of the number of instances of the abstract class relations

Abstract class relation	Minimal required instances A	B	Comparison number of A / number of B	Abstract class relation	Minimal required instances A	B	Comparison number of A / number of B
1. $R_{LD.RD}$	2	2	-	10. $R_{RD.RT}$	3	1	A > B + 1
2. R_{LD}	2	3	-	11. R_{LT}	2	3	-
3. R_{RD}	3	2	-	12. R_{RT}	3	2	-
4. R_{some}	3	3	-	13. $R_{LD.RD.LT.RT}$	2	2	A = B
5. $R_{LD.RD.LT}$	1	2	A < B	14. $R_{LD.LT.RT}$	2	3	A < B
6. $R_{LD.RD.RT}$	2	1	A > B	15. $R_{RD.LT.RT}$	3	2	A > B
7. $R_{LD.RT}$	2	2	-	16. $R_{LT.RT}$	2	2	-
8. $R_{RD.LT}$	2	2	-	17. $R_{LT.RT\text{-}all}$	1	1	-
9. $R_{LD.LT}$	1	3	A + 1 < B				

Please note that class relations are not depending on fixed numbers of instances and the constellations represented in figure 2 are just exemplarily. Some abstract class relations require a minimum number of instances of A and/or B and a certain ratio between the instances of both classes. These restrictions are represented in table 1.

The set of the 17 abstract class relations is a qualitative representation of the constellation of instance relations between two classes. For every instance relation there is only one class relation valid for each pair of classes. Nevertheless it is possible to define more than one class relation between two classes, even when the applied instance relations are part of the same jointly exhaustive and pair wise disjoint (JEPD)

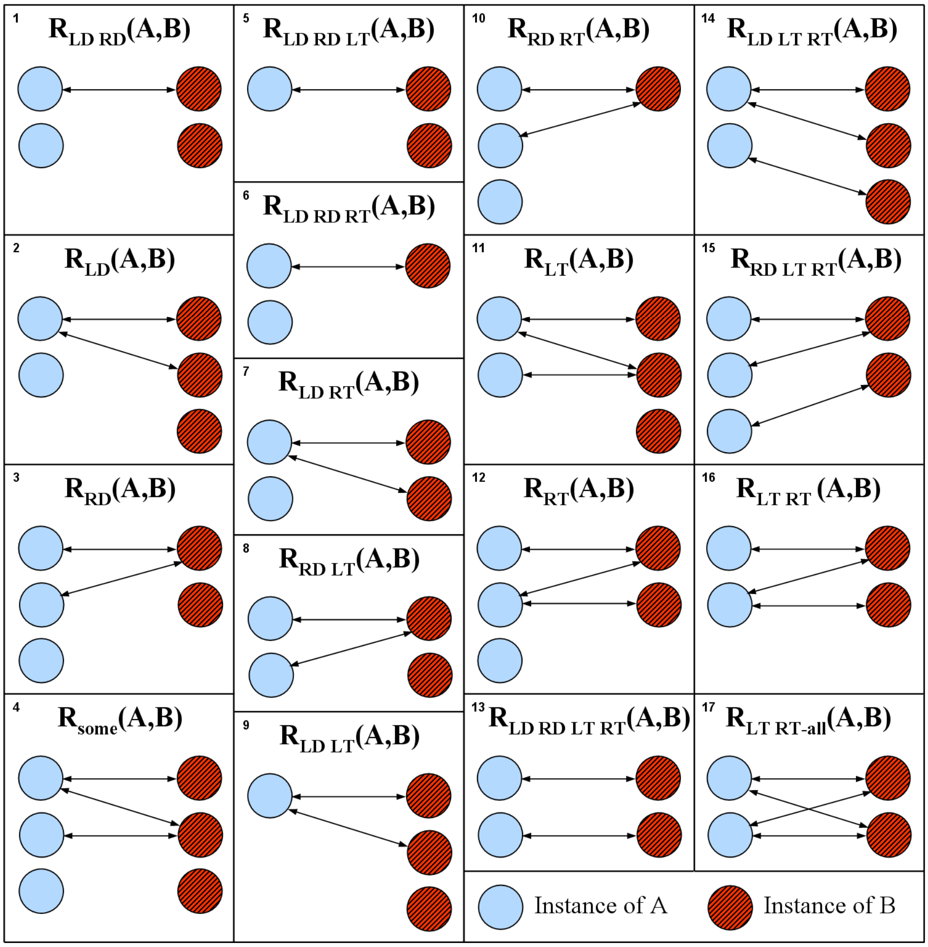


Fig. 2. Examples of the 17 abstract class relations

set of instance relations. Based on the definitions of the cardinality properties and the abstract class relations [11] it can be proven that for a JEPD set of instance relations the corresponding class relations are also JEPD (hence each class relation definition excludes the cardinality properties which are not valid, see (CR1)).

3 Reasoning on Class Relations

The reasoning methods presented in this chapter can be used to find conflicts and redundancies in sets of class relations. As demonstrated in [1] and [11] the logical properties of class relations derive from the logical properties of the corresponding instance and abstract class relations. It has been shown that it is appropriate to analyse the reasoning properties of the abstract class and instance level relations independently of each other. For the abstract class relations it is reasonable to refer the logical properties to their cardinality definitions. The following sections investigate the symmetry, composition and conceptual neighbourhood of the 17 abstract class relations.

3.1 Symmetry of Class Relations

Logical properties class relations, like symmetry and transitivity, have been researched in [1]. As pointed out by [11] in particular the symmetry is of interest, since this property has to be proven to ensure the arc consistency of a class relation network. It has been shown that every class relation has an inverse relation, if the corresponding instance relation has an inverse relation or is symmetric. Most spatial relations fulfil this requirement. The symmetry properties of the class relations can be derived from the symmetry of the applied instance relations and the cardinality definitions of the abstract class relations ((CP1)-(CP5), (CR2) and (CR3)). The inverse of a class relation is also based on the inverse of the applied instance relation. If an abstract class relation is *left-total* / *left-definite* the inverse relation is *right-total* / *right-definite* and vice versa. R_{some} and $R_{LT.RT\text{-}all}$ are symmetric. The following two examples demonstrate the derivation of inverse class relations. Here the class relations are based on the symmetric instance relation *disjoint* and the inverse relations *contains* and *inside*:

$$(\text{DISJOINT}_{\text{RD.LT}}(A,B))^{i} = \text{DISJOINT}_{\text{LD.RT}}(B,A).$$
$$(\text{CONTAINS}_{\text{LD.RD.LT}}(A,B))^{i} = \text{INSIDE}_{\text{LD.RD.RT}}(B,A).$$

3.2 Composition of Class Relations

The composition of binary relations enables the derivation of implicit knowledge about a triple of entities. If two binary relations are known the corresponding third one can potentially be inferred or some of the possible relations can be excluded. This knowledge can also be used to find conflicts in case all of the three relations are known. The compositions rules of a set of relations are usually represented in a composition table like it has been done for the topological relations between areal entities in [4][8] and for directional/orientation relations in [6][9]. Many other sets of binary spatial relations also allow for such derivations.

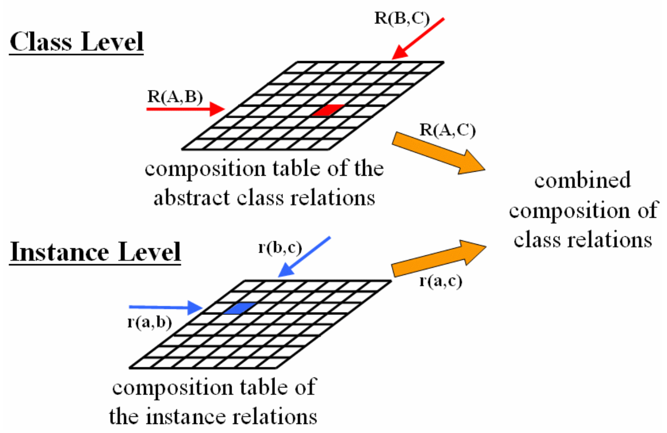

Class Level
R(B,C)
R(A,B)
R(A,C)
composition table of the abstract class relations
combined composition of class relations
Instance Level
r(b,c)
r(a,b)
r(a,c)
composition table of the instance relations

Fig. 3. Two levels composition of class relations

The composition of class relations is hardly researched yet, but as shown in [11] it is also possible. This paper proposes a two level reasoning formalism, which separates the compositions of the abstract class relations from those of the instance relations (see figure 3). Therewith the composition of the abstract class relations can be defined independently of a concrete set of instance relations.

The following example demonstrates how the compositions of the abstract class relations are inferred. With the class relations $R1_{LT.RT\text{-}all}(A,B)$ and $R2_{LD.RD.LT.RT}(B,C)$ given, the relation between the classes A and C shall be derived. Such a situation is schematically represented in figure 4. All instances of A have the same instance relation r1 to all instances of B and all instances of B are related by r2 to one instance of C. To infer the composition of the abstract class relations every possible triple of A, B and C instances has to be separately analysed. Whenever the relation r1 between the instance of A and the instance of B and the relation r2 between the instance of B and the instance of C is given, the relation r3 (or a disjunction of possible relations) between the A and C instances can be inferred. The combination of the inferences of all possible triples of instances leads to the abstract class relation between A and C. If no triple of instances with r1 and r2 relations exists, then no inference for r3 is possible. Therewith the composition of the class relations leads to a universal disjunction $\mathcal{U}$ of all possible class relations. For the example shown in figure 4, each instance of A is related to every instance of C via some instance of B. Therewith it is obvious that all instances of A must have the same instance relation to all instances of C. Thus the composition of the abstract class relations must be:

$$R1_{LT.RT-all}(A,B); R2_{LD.RD.LT.RT}(B,C) \Rightarrow R3_{LT.RT-all}(A,C).$$

For the composition of the class relations this result must be combined with the composition of the instance relations, for example (taken from the composition table in [8]):

$$\text{meet}(a,b); \text{covers}(b,c) \Rightarrow \text{disjoint}(a,c) \cup \text{meet}(a,c).$$

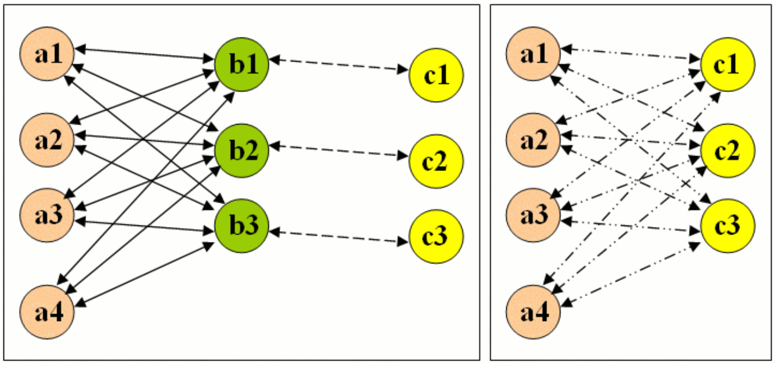


Fig. 4. Possible scene defined by the class relations $R1_{LT.RT\text{-}all}(A,B)$ and $R2_{LD.RD.LT.RT}(B,C)$ and their composition $R3_{LT.RT\text{-}all}(A,C)$

The combination of the compositions of the two levels results in:

$$\mathrm{MEET}_{\mathrm{LT.RT-all}}(\mathrm{A,B});\mathrm{COVERS}_{\mathrm{LD.RD.LT.RT}}(\mathrm{B,C}) \Rightarrow [\mathrm{DISJOINT} \cup \mathrm{MEET}]_{\mathrm{LT.RT-all}}(\mathrm{A,C}).$$

For the example shown in figure 4 the derived composition is independent of the number of instances. This means that the composition of the given abstract class relations will always lead to the same result. The 17 abstract class relations have 289 possible compositions. Many of them have differing results, depending on the number of instances of the three classes and the relative arrangement of the instance relations. This must be considered when the compositions are calculated.

The influence of the relative arrangement of the instance relations on the composition is illustrated in figure 5. The two boxes show possible constellations of the $R1_{LT.RT}$; $R2_{LT.RT}$ composition. They only differ in an instance relation between the classes B and C: in the left box b1 and c2 are related whereas in the right box the instances b2 and c1 are related. The abstract class relations and the total amount of instance relations are the same. Nevertheless this difference will lead to different compositions. For the left constellation the relation between the instances a2 and c1 can not be inferred and the composition is $R3_{LT.RT}$ (relation #16). The right constellation allows for a deduction of all four instance relations between A and C and thus the composition is $R_{LT.RT\text{-}all}$ (relation #17). The disjunction of all possible results leads to the composition of the abstract class relations:

$$\mathrm{R1}_{\mathrm{LT.RT}}(\mathrm{A,B});\mathrm{R2}_{\mathrm{LT.RT}}(\mathrm{B,C}) \Rightarrow \mathrm{R3}_{\mathrm{LT.RT}}(\mathrm{A,C}) \cap \mathrm{R3}_{\mathrm{LT.RT-all}}(\mathrm{A,C}).$$

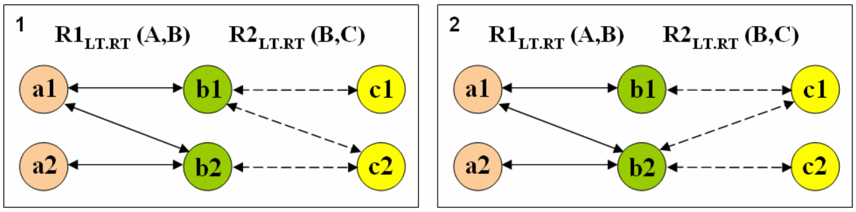


Fig. 5. Example of how the relative arrangement of the instance relations influences the composition of the abstract class relations

In previous publications the composition of class relations has only been exemplarily investigated; an overall analysis of all possible compositions has never been presented. The calculation of this composition table is complex and costly. For the defined abstract class relations it requires an analysis of all possible arrangements of instance relations for up to 6 instances for each of the three classes. If both classes of one relation have 6 instances than there are about 68,7 billion arrangements possible. Each of these has to be separately analysed with all possible arrangements of the second relation. An analysis of classes with 7 or more instances does not lead to additional results in the composition. Making use of some heuristics this calculation can be further optimised. The overall composition table is shown in figure 6.

2. Relation / 1. Relation		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	LD.RD	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	1	$\mathcal{U}$	$\mathcal{U}$	1,3	1,2	$\mathcal{U}$	1,2, 3,4	$\mathcal{U}$	1	1,2	1,3, 6,10	1,2,3, 4,6,7, 10,12	6, 7, 10,12
2	LD	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	2	$\mathcal{U}$	$\mathcal{U}$	1,2, 3,4	2	$\mathcal{U}$	1,2, 3,4	$\mathcal{U}$	2	2	1,2,3,4, 6,7,10, 12	1,2,3, 4,6,7, 10,12	6, 7, 10,12
3	RD	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	3	$\mathcal{U}$	$\mathcal{U}$	3	3,4	$\mathcal{U}$	3,4	$\mathcal{U}$	3	3,4	3,10	3,4, 10,12	10,12
4	some	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	4	$\mathcal{U}$	$\mathcal{U}$	3,4	4	$\mathcal{U}$	3,4	$\mathcal{U}$	4	4	3,4, 10,12	3,4, 10,12	10,12
5	LD.RD.LT	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	5	$\mathcal{U}$	$\mathcal{U}$	5,8	5,9	$\mathcal{U}$	5,8, 9,11	$\mathcal{U}$	5	5,9	5,8, 13,15	5,8,9, 11,13,14, 15,16,17	17
6	LD.RD.RT	1	2	3	4	1	6	7	3	2	10	4	12	6	7	10	12	6, 7, 10,12
7	LD.RT	1,2	2	1,2, 3,4	2,4	2	6,7	7	1,2, 3,4	2	6,7, 10,12	2,4	7,12	7	7	6,7, 10,12	7,12	6, 7, 10,12
8	RD.LT	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	8	$\mathcal{U}$	$\mathcal{U}$	8	8,11	$\mathcal{U}$	8,11	$\mathcal{U}$	8	8,11	8,15	8,11,15, 16,17	17
9	LD.LT	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	9	$\mathcal{U}$	$\mathcal{U}$	5,8, 9,11	9	$\mathcal{U}$	5,8, 9,11	$\mathcal{U}$	9	9	5,8,9,11, 13,14,15 16,17	5,8,9,11, 13,14,15, 16,17	17
10	RD.RT	1,3	2,4	3	4	3	6,10	7,12	3	4	10	4	12	10	12	10	12	10,12
11	LT	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	11	$\mathcal{U}$	$\mathcal{U}$	8,11	11	$\mathcal{U}$	8,11	$\mathcal{U}$	11	11	8,11, 15,16, 17	8,11, 15,16, 17	17
12	RT	1,2 3,4	2,4	1,2, 3,4	2,4	4	6,7, 10,12	7,12	3,4	4	6,7, 10,12	4	7,12	12	12	10,12	12	10,12
12	RT	1,2 3,4	2,4	1,2, 3,4	2,4	4	6,7, 10,12	7,12	3,4	4	6,7, 10,12	4	7,12	12	12	10,12	12	10,12
13	LD.RD.LT.RT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
14	LD.LT.RT	1,2, 5,9	2,9	1,2,3, 4,5,8, 9,11	2,4, 9,11	9	6,7, 13,14	7,14	5,8, 9,11	9	6,7,10, 12,13,14, 15,16,17	9,11	7,12, 14,16, 17	14	14	13,14, 15,16, 17	14,16, 17	17
15	RD.LT.RT	1,3	2,4	3	4	8	6,10	7,12	8	11	10	11	12	15	16	15	16	17
16	LT.RT	1,2,3, 4,5,8, 9,11	2,4, 9,11	1,2,3, 4,5,8, 9,11	2,4, 9,11	11	6,7,10, 12,13,14, 15,16,17	7,12, 14,16, 17	8,11	11	6,7,10, 12,13,14, 15,16,17	11	7,12, 14,16, 17	16	16	15,16, 17	16,17	17
17	LT.RT-all	5,8, 9,11	9,11	5,8, 9,11	9,11	5,8, 9,11	17	17	5,8, 9,11	9,11	17	9,11	17	17	17	17	17	17

Fig. 6. Composition table of the 17 abstract class relations

Some of the compositions can be summarized by general rules, which deduce the composition directly from the cardinality properties. This allows a more convenient use of the composition. Some obvious rules are:

– If the first abstract class relation is not *right-total* and the second relation is not *left-total* the composition is always a universal disjunction $\mathcal{U}$.
– If the first relation is $R_{LD.RD.RT}$ (relation #6) and the second is not *left-total* the composition is equal to the second abstract class relation.
– If the first relation is $R_{LD.RD.RT}$ (relation #6) and the second is *left-total* the composition has the same cardinality properties as the second relation, but it is not *left-total.* For $R_{LT.RT\text{-}all}$ (relation #17) this can be relation #6, #7, #10 or #12.

- If the first relation is not *right-total* and the second is $R_{LD.RD.LT}$ (relation #5) the composition is equal to the first abstract class relation.
- If the first relation is *right-total* and the second is $R_{LD.RD.LT}$ (relation #5) the composition has the same cardinality properties as the first relation, but it is not *right-total.* For $R_{LT.RT\text{-}all}$ (relation #17) this can be relation #5, #8, #9 or #11.
- If one of the relations is $R_{LD.RD.LT.RT}$ (relation #13) the composition is always equal to the corresponding other abstract class relation. Because of this property $R_{LD.RD.LT.RT}$ can represent the identity relation of classes if it is combined with a identity instance relation, e.g. $EQUAL_{LD.RD.LT.RT}(A,A)$.
- If the first relation is $R_{LT.RT\text{-}all}$ (relation #17) and the second is *right-total* the composition is always $R_{LT.RT\text{-}all}$.
- If the first relation is *left-total* and the second is $R_{LT.RT\text{-}all}$ (relation #17) the composition is always $R_{LT.RT\text{-}all}$.

The compositions which are defined by these rules are highlighted in grey in figure 6. A set of rules which completely represents the composition table is a subject of further research.

For the abstract class relations and their composition table the properties of a relation algebra [12] have been computationally checked. Some properties of the presented composition of abstract class relations are:

- The inverse of an inverse relation is equal to the original relation: $(R^i)^i = R$.
- All compositions with the identity relation (relation #13) are idempotent: $R;R_{LD.RD.LT.RT} \rightarrow R$ and $R_{LD.RD.LT.RT};R \rightarrow R$.
- The inverse of a composition is equal to the composition of the inverses of the two relations in reverse order: $(R1\ ;\ R2)^i = R2^i\ ;\ R1^i$.
- The associative property $(R1;R2);R3 = R1;(R2;R3)$ and the semiassociative property $R;\ (\mathcal{U};\mathcal{U}) = (R\ ;\mathcal{U});\mathcal{U}$ [10] are not valid. Therewith the composition of the abstract class relations is nonassociative.

3.3 Conceptual Neighbourhood of Class Relations

The conceptual neighbourhood represents continuous transformations between relations through linking relations that are connected by an atomic change. [6] defines two relations in a representation as conceptual neighbours, "if an operation in the represented domain can result in a direct transition from one relation to the other." Examples of conceptual neighbourhood networks of instance relations can be found for temporal interval relations in [7], for topological relations between regions in [3] and between regions and lines in [5].

The conceptual neighbourhood of class relations has not yet been researched. In this approach two class relations are considered as conceptually neighboured if they are linked to the same instance relation and only differ by a single instance relation between two entities. The number of instances of the class is considered as fixed. In figure 7 the conceptual neighbourhood of R_{some} and $R_{LT.RT}$ is exemplarily illustrated. All arrows symbolize one instance relation of the same kind *r*. The addition of a further instance relation, represented by the dashed arrow in the right box, leads to a transition of the abstract class relation from R_{some} to $R_{LT.RT}$.

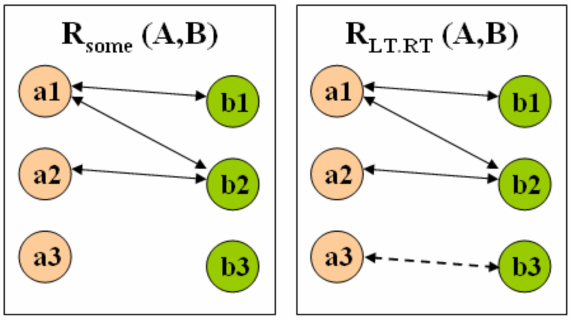


Fig. 7. Conceptual neighbourhood between R_{some} and $R_{LT.RT}$

Table 2. Conceptual neighbourhood of the class relations; **+/-** represents neighbourhood through addition/removal of an instance relation

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1 LD.RD		+	+	+	+	+	+	+					+				
2 LD	-			+			+		+		+			+			
3 RD	-			+				+		+		+			+		
4 some	-	-	-								+	+				+	
5 LD.RD.LT	-								+		+			+			+
6 LD.RD.RT	-									+		+			+		+
7 LD.RT	-	-										+				+	
8 RD.LT	-		-								+					+	
9 LD.LT		-			-						+			+			+
10 RD.RT			-			-						+			+		+
11 LT		-		-	-			-	-							+	
12 RT			-	-		-	-			-						+	
13 LD.RD.LT.RT	-															+	
14 LD.LT.RT		-			-				-							+	
15 RD.LT.RT			-			-				-						+	
16 LT.RT				-			-	-			-	-	-	-	-		+
17 LT.RT-all					-	-			-	-						-	

The computation of all conceptual neighbourhoods between class relations requires an analysis of all possible arrangements of instance relations for up to 4 instances for both classes. A higher number of instances does not lead to additional results. For the 17 class relations are all together 45 neighbourhoods existing. Since the neighbourhood is defined through adding or removal of a single instance relation all neighbourhoods are directed. Table 2 represents the neighbourhoods which result from an addition by a "+" and those which result from a removal by a "-". If a class relation has changed though an addition / removal of an instance relation, it is not possible to get the same class relation again by further adding / removing of instance relations. The adding of instance relations will ultimately lead to $R_{LT.RT\text{-}all}$ (relation #17). A removal will lead to $R_{LD.RD}$ (relation #1)[1]. The numbering of the abstract class relations

[1] For this relation both involved classes must have at least two entities, see table 1.

has been chosen such that for all class relations the relations which result from an addition have a higher number and all inverse relations are successive. Because of this order all removal neighbourhoods appear at the left bottom and all addition neighbourhoods at the right top in table 2.

The following example shall illustrate the practical use of the conceptual neighbourhood of class relations. Three class relations are defined for the classes A, B and C: *MEET$_{some}$(A,B)*, *CONTAINS$_{LD.RD.LT.RT}$(B,C)* and *DISJOINT$_{LT}$(A,C)*. These relations shall be analysed for conflicts through comparing the composition of the class relations A to B and B to C with the given third relation between A and C. The compositions of the corresponding instance and abstract class relations are:

$$\text{meet}(a,b);\text{contains}(b,c) \Rightarrow \text{disjoint}(a,c).$$
$$\text{R1}_{\text{some}}(A,B);\text{R2}_{\text{LD.RD.LT.RT}}(B,C) \Rightarrow \text{R3}_{\text{some}}(A,C).$$

Thus the combination of the compositions of the two levels results in:

$$\text{MEET}_{\text{some}}(A,B);\text{CONTAINS}_{\text{LD.RD.LT.RT}}(B,C) \Rightarrow \text{DISJOINT}_{\text{some}}(A,C).$$

This result seems to be in conflict to the given third relation *DISJOINT$_{LT}$(A,C)*. Figure 8 exemplarily illustrates this situation. The first box shows the given class relations and the second the inferred relation between A and C. In comparison with this the third box shows that the given relation *DISJOINT$_{LT}$(A,C)* possibly differs from *DISJOINT$_{some}$(A,C)* by only one *disjoint* instance relation (in this case a3 to c2). Therewith *DISJOINT$_{some}$(A,C)* and *DISJOINT$_{LT}$(A,C)* are conceptual neighbours. In figure 8 the three *disjoint* instance relations of *DISJOINT$_{LT}$(A,C)* are implied by the class relations A to B and B to C. About further relations between the instances of A and C the composition does not allow for any conclusion. It can also not be excluded that further pairs of A and C instances are *disjoint*. Hence the composition of *MEET$_{some}$(A,B)* and *CONTAINS$_{LD.RD.LT.RT}$(B,C)* does not conflict *DISJOINT$_{LT}$(A,C)* and the triple of class relation is consistent. Beside *DISJOINT$_{LT}$(A,C)* also the class relations *DISJOINT$_{RT}$(A,C)* and *DISJOINT$_{LT.RT}$(A,C)* as direct conceptual neighbours of *DISJOINT$_{some}$()* and *DISJOINT$_{LT.RT-all}$(A,C)* as conceptual neighbour of *DISJOINT$_{LT.RT}$()* would not conflict.

In general, a class relation R3 is not in conflict with a composition R1 ; R2 → R3* if R3* and R3 are based on the same instance relation r3 and the addition of further r3 instance relations to R3* can lead to a transition to class relation R3. For this the result of the composition R3* and R3 don't need to be direct conceptual neighbours. There can also be further class relation transitions between the two class relations. Nevertheless the conceptual neighbourhood points out which R3 class relations are valid, since it shows which transitions are possible for a certain class relation R3*.

Thus the check of conflicts in a triple of class relations consists of two steps: first the comparison of the composition of two relations with the given third. If they are equal the triple of relations is conform to the introduced composition of class relations and there is no obvious conflict. If these two relations are not equal the second step checks their conceptual neighbourhood as described above. If the given third relation is not a corresponding conceptual neighbour of the composition the triple of class relations is conflicting.

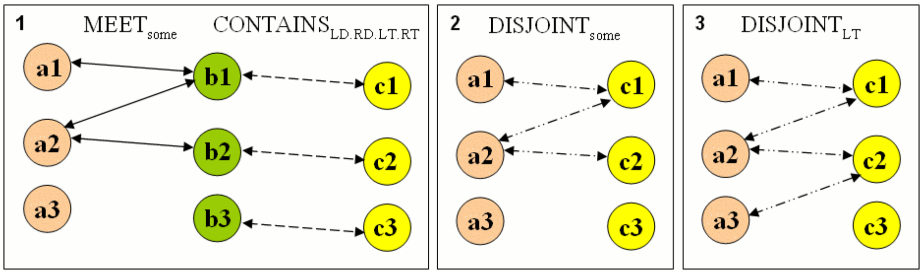


Fig. 8. Use of the conceptual neighbourhood for the composition of class relations

4 Conclusion and Open Issues

The scientific investigation of class relations is currently still in the early stages. This work continues research into spatial class relation by deepening the analysis of the reasoning properties of the class relations. It is based on a set of 17 abstract class relations defined in [11]. The paper focuses on the composition and conceptual neighbourhood of class relations. The definitions and reasoning rules of the class relations are described independently of a specific set of instance relations. The introduced two levels composition of class relations allows for a separate analysis of instance relations and abstract class relations. Therewith the overall reasoning formalism can be used with any spatial or non-spatial set of instance relations. The only requirements imposed on the instance relations are that they are part of a JEPD set of relations and have defined inverse relations and compositions.

With the described logics it is possible to find conflicts and redundancies in networks of class relations. This can for example be applied to prove consistency of sets of spatial semantic integrity constraints or spatial relations between classes in an ontology.

This approach is restricted to binary relations between entire entity classes. Relations between three or more classes or between subsets of classes are not considered. Further more, only total participation and a cardinality ratio of 0..1 are included as cardinality properties of the class relations. Nevertheless this framework provides a basis, which can be extended for other possibly more complex types of class relations. For an extension by further cardinality ratio constraints (e.g. 0..2) it has to be considered, that this will increase the calculation cost of the compositions exponentially.

Further work can also deal with the direct derivation of the reasoning properties of the class relations from their cardinality properties. This will deepen the understanding of the logics and support possible extensions by additional cardinality properties. The two levels composition of class relations separates the compositions of the abstract class relations from those of the instance relations (figure 3). However, some combinations of instance and abstract class relations lead to conflicts which can not be found this way. For example the combination of $EQUAL_{LD.RT}(A,B)$ and $R_{LD.RD.LT.RT}(B,C)$ (see figure 2: relation #7; relation #13) is not possible. This is due to the specific properties of the *equal* identity instance relation and the cardinality properties of the two abstract class relations. A general description of such conflicts is unsolved.

As pointed out in table 1 some abstract class relations require a minimum number of instances in A and/or B and a certain ratio between the instances of both classes. For many entity classes the number of existing individuals is unknown or variable. For these classes the restriction of the number of individuals is irrelevant. However, for classes with a small and well defined number of instances (e.g. earth surface or continents) the designer of a data model or an ontology is in many cases aware of these numbers. The knowledge about these numbers and their correlation to the class relations can be included in reasoning about class relations. The restrictions on the number of instances also lead to restrictions of the composition and the conceptual neighbourhood.

References

1. Donnelly, M., Bittner, T.: Spatial Relations Between Classes of Individuals. In: Cohn, A.G., Mark, D.M. (eds.) COSIT 2005. LNCS, vol. 3693, pp. 182–199. Springer, Heidelberg (2005)
2. Egenhofer, M., Herring, J.: Categorizing Binary Topological Relationships Between Regions, Lines, and Points in Geographic Databases. Technical Report, Department of Surveying Engineering. University of Maine, Orono, ME (1991)
3. Egenhofer, M., Al-Taha, K.K.: Reasoning about Gradual Changes of Topological Relationships. In: Frank, A.U., Formentini, U., Campari, I. (eds.) GIS 1992. LNCS, vol. 639, pp. 196–219. Springer, Heidelberg (1992)
4. Egenhofer, M.: Deriving the Composition of Binary Topological Relations. Journal of Visual Languages and Computing 5(2), 133–149 (1994)
5. Egenhofer, M., Mark, D.: Modeling Conceptual Neighborhoods of Topological LineRegion Relations. International Journal of Geographical Information Systems 9, 555–565 (1995)
6. Freksa, C.: Using Orientation Information for Qualitative Spatial Reasoning. In: Frank, A.U., Formentini, U., Campari, I. (eds.) GIS 1992. LNCS, vol. 639, pp. 162–178. Springer, Heidelberg (1992)
7. Freksa, C.: Temporal reasoning based on semi-intervals. Artificial Intelligence 54, 199–227 (1992)
8. Grigni, M., Papadias, D., Papadimitriou, C.: Topological inference. In: Proceedings of the International Joint Conference of Artificial Intelligence (IJCAI), pp. 901–906 (1995)
9. Hernandez, D.: Qualitative Representation of Spatial Knowledge. In: Hernández, D. (ed.) Qualitative Representation of Spatial Knowledge. LNCS, vol. 804. Springer, Heidelberg (1994)
10. Maddux, R.: Some varieties containing relation algebras. Transactions of the American Mathematical Society 272(2), 501–526 (1982)
11. Mäs, S.: Reasoning on Spatial Semantic Integrity Constraints. In: Winter, S., Duckham, M., Kulik, L., Kuipers, B. (eds.) COSIT 2007. LNCS, vol. 4736, pp. 285–302. Springer, Heidelberg (2007)
12. Tarski, A.: On the Calculus of Relations. The Journal of Symbolic Logic 6(3), 73–89 (1941)

Identifying Maps on the World Wide Web

Matthew Michelson, Aman Goel, and Craig A. Knoblock*

University of Southern California,
Information Sciences Institute,
Marina del Rey, CA 90292, USA
{michelso,amangoel,knoblock}@isi.edu
http://www.isi.edu/integration

Abstract. This paper presents an automatic approach to mining collections of maps from the Web. Our method harvests images from the Web and then classifies them as maps or non-maps by comparing them to previously classified map and non-map images using methods from Content-Based Image Retrieval (CBIR). Our approach outperforms the accuracy of the previous approach by 20% in F_1-measure. Further, our method is more scalable and less costly than previous approaches that rely on more traditional machine learning techniques.

1 Introduction

As more and more information comes online, disciplines ranging from medicine to geography benefit from the proliferation of freely available data. For example, in the field of geography, huge repositories of maps can be built by harvesting maps from all of the images on the Web. Once harvested, these maps can then be aligned to known geographic sources in a process known as "georeferencing" [1,2,3]. Previous work georeferences maps to satellite images by first automatically extracting the roads and intersections from the map image [4,5,6], and then using those roads and intersections to automatically align the map to the satellite image in a process known as conflation [1,2]. Once georeferenced, the map collection can be queried using the same spatial queries used on the satellite images, such as queries by region, latitude/longitude, street name, etc. Further, not only are the maps returned for the given query, but they can be put in context by overlaying the map images on top of the satellite image. An example of a conflated map with a satellite image is given in Figure 1.

* This research is based upon work supported in part by the National Science Foundation under award number IIS-0324955, in part by the United States Air Force under contract number FA9550-08-C-0010, and in part by a gift from Microsoft. The U.S. Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

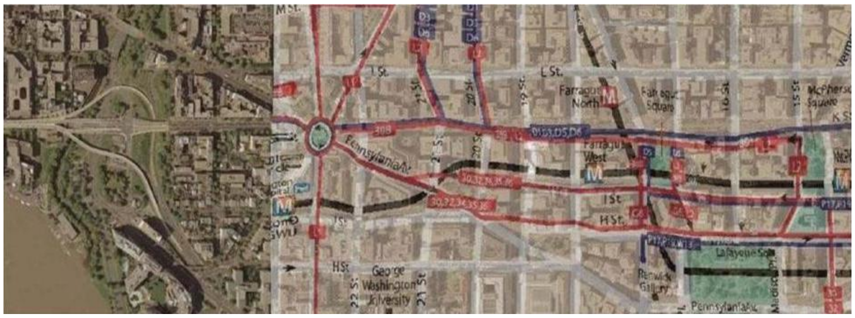


Fig. 1. A bus map for Washington, DC conflated with a satellite image of the city

In this paper we focus on the first of these problems, automatically harvesting maps from the freely available images on the Web. However, given that there is no filter for publishing on the Web, a major challenge exists in separating the map images from the other types of images. Manually sifting through the images is both tedious and costly, especially given the huge amounts of data that must be examined. In this paper we address the problem of *automatically* separating geographic maps from other images collected on the World Wide Web. An autonomous approach not only eases the cost associated with identifying maps, but it also provides an easy and scalable approach to growing a collection of maps over time. Once collected, such maps are useful for data integration, intelligence analysis and geographic information systems. While we focus on the

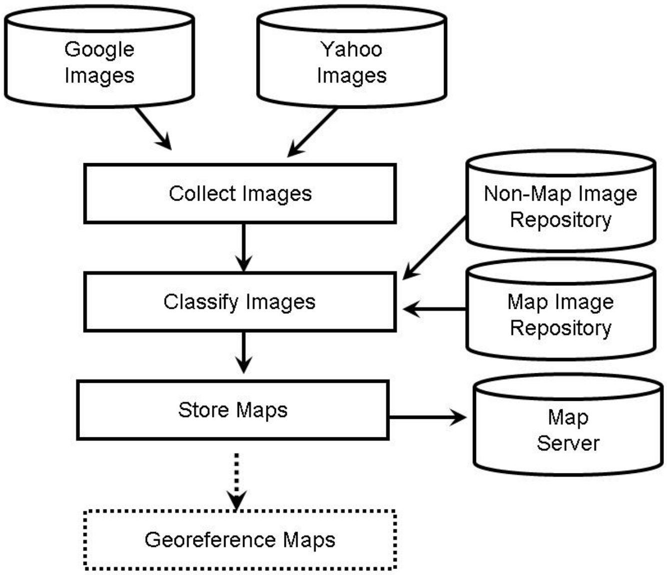


Fig. 2. A software system for automatically collecting maps from the Web

particular task of identifying maps, we believe the approach is general enough to work for other specific image types, such as medical images.

More specifically, we start with a collection of images harvested from the Web and our goal is to sort the images into maps and other images. Rather than collecting the images ourselves, which is costly, we leverage the image-search databases of large companies such as Microsoft, Yahoo!, and Google who index and collect millions of images from all over the Web. Since these databases are refreshed frequently with new content, this allows us to collect new maps over time without having to spider the Web itself. Using such image databases, we can collect the search results and classify each returned image as a map or non-map, storing the newly classified maps in a map server. These stored maps can then be georeferenced for future querying. Figure 2 shows this process.

As shown in Figure 2, we exploit two repositories, one of maps and one of non-maps, to do our classification. While the specifics of our classifier are given in Section 2, intuitively, if an image to be classified is more similar to the maps in our repository than the non-maps, then it is more likely a map itself. That is, using techniques from Content-Based Image Retrieval (CBIR), we select the most similar images from the repositories for a given map and use those returned images to decide if the input image is a map or not. We find that our CBIR method is both more scalable than machine learning methods (as justified in Section 2), and it is also more accurate (as shown in the experiments of Section 3). This is the basis of our contribution: a method to sort images into maps that is scalable, robust, autonomous and accurate.

The rest of this paper is organized as follows. We describe the details of our CBIR map classifier in Section 2. We then describe our experiments and justify our approach in Section 3. We next present related work in Section 4, and we finish with our conclusions and future research in Section 5.

2 Automatically Classifying Maps

The focus of this paper is the classification of maps from images harvested from the Web. In this paper we broadly define a map as any digital image which contains "map like" characteristics such as lines, intersections and labels. The important aspect for our task is that we remain consistent in our definition of maps versus non-maps.

Our map classifier exploits techniques from Content-Based Image Retrieval (CBIR).[1] CBIR is essentially the image equivalent to traditional text based Web search. Instead of finding the most similar Web pages for a given set of query terms, in CBIR the most similar images from a set are returned for a given query image. For example, consider Figure 3. In this figure, the query image comes into the system and the top three most similar images are returned. This is the basis of our classifier.

To exploit CBIR for classification, we use a voting, k-Nearest Neighbor classifier [8]. We first use CBIR to find the nine most similar images (neighbors)

[1] See [7] for an excellent survey of the topic

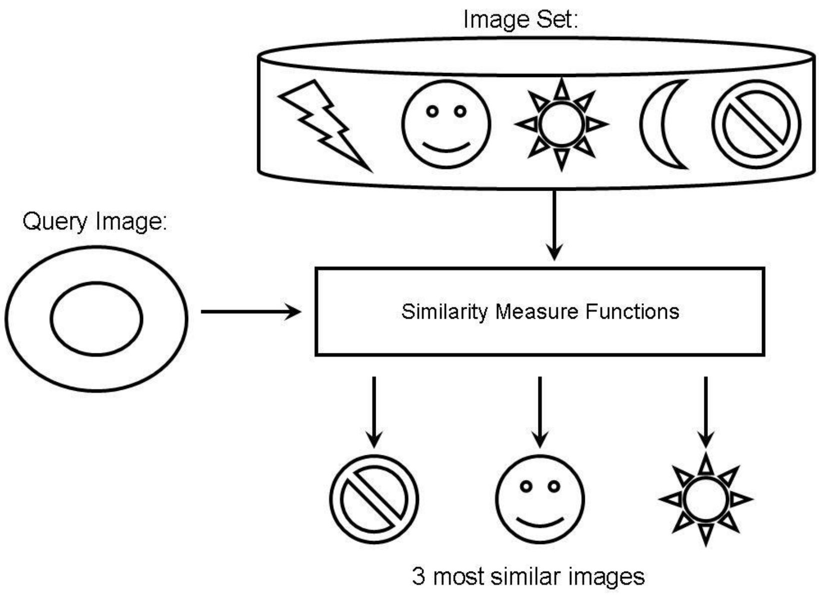


Fig. 3. An example of Content-Based Image Retrieval

from the combined set of images in the map and non-map repositories. The similarity measure in our CBIR method is based on Water-Filling features [9]. Water-filling uses the edge maps of an image to quantify measures such as edge complexity (based on forking), edge lengths, etc. These Water-Filling features are well suited for images with clear edge structures [9], such as maps. Further, by using edge-based features we make our classifier color invariant.

We then employ a simple majority voting mechanism [8]. If the majority of returned images are maps, we classify the query image as a map. If the majority of returned images are non-maps, we classify the query image as a non-map. This simple algorithm is shown in Figure 4. Therefore, although it may be the case that other images on the Web will have clear edge structures (such as diagrams), since our technique relies not only on the edge features themselves, but also on the similarity to the edge features of the images in our map repository, such images will be filtered out. The accuracy of our experimental results indeed show this to be the case.

We choose a CBIR based k-Nearest Neighbor classifier over more traditional machine learning methods for the following reasons. First, CBIR similarity methods allow us to exploit image similarities without explicitly modeling them. For instance, hydrography maps are similar to other hydrography maps and urban city maps are more similar to urban city maps but these types of maps may be quite different from each other. By using CBIR methods, these similarities can be exploited without modeling them explicitly because the returned images encompass the image similarities implicitly (that is why they are returned as similar). If we use traditional machine learning methods, such as Support Vector Machines, we have two options to capture these different types of image

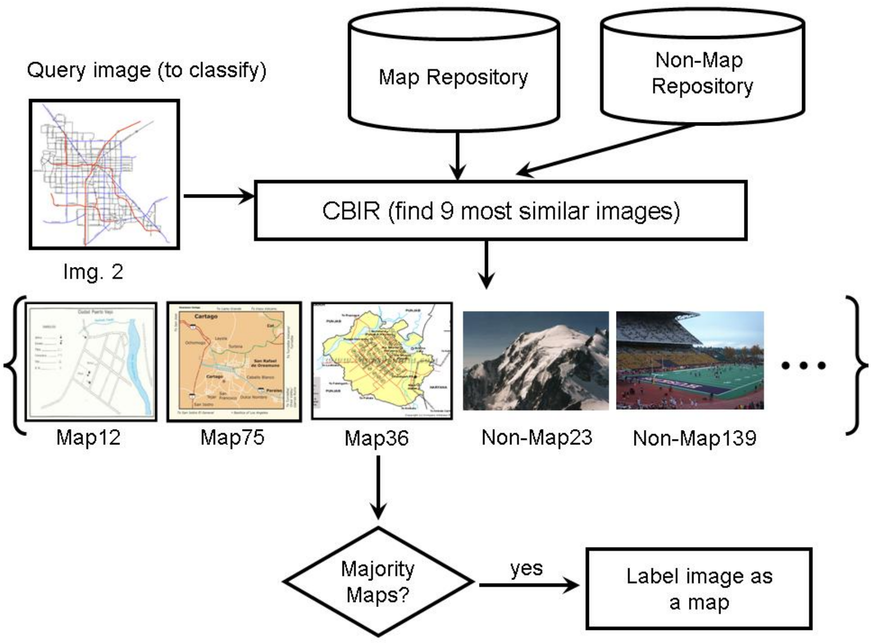


Fig. 4. A k-Nearest Neighbor map classifier using CBIR

similarity. On the one hand, we can train a model for each class of map, then if an incoming image matches any of these map classes, we know it is a map (since each class composes the image similarities). That is, we can take the hydography maps and learn a model of what a hydrography map should be. We can then take the urban city map and learn an urban city map model.

There are several problems with trying to learn a model for each type of map. For one, the number of such classes is not known in advance. Therefore, a user will have to make a decision as to which maps constitute a class and hope he or she made the correct choices to lead to accurate classification. Along these lines, the user must make decisions as to the granularity of the classes (is an urban-hydrographical map its own class?), which can further complicate the creation of classes. Also, learning a new model may be a non-trivial process both in the work and time required. So, if a new class is discovered or needed, this can become a prohibitively costly problem.

Instead, rather than modeling all of the different types of image similarities in distinct classes, one could try to learn a single model that encompasses *all* of these different types of similarities. However, since different types of map images can vary wildly, trying to generalize to cover all of these different similarities leads to a learned model that does not discriminate well (as shown in our experiments).

The other reason we chose CBIR instead of machine learning has to do with robustness and scalability. We can exploit huge repositories in our method, which is not the case using machine learning. In machine learning, learning models from massive amounts of data can take so long that it is not practical. CBIR techniques, however, are built on methods from information retrieval which the major search engines have shown to be fast and robust in very large, practical settings. Further, we can freely tweak the size and composition of our repository to test the effect (something we do in the experiments to test this idea). Using machine learning, we have to retrain a model each time we tweak the repository (training data). In situations where training is costly, this is not a good solution. Therefore, by using CBIR methods we can grow the repository over time without retraining which allows for a scalable and autonomous solution to classifying maps and building good map repositories. In our experiments, we show the effects of growing the repository on the accuracy of classification.

3 Experiments

We collected 1,704 map images and 3,462 non-map images for our experiment. We used Yahoo Image Search[2] to gather all of the map images and some of the non-map images. Most of the non-map images come from the CALTECH 101 data set [10], which is a set of pictures of objects belonging to 101 different categories. Table 1 presents the distribution of images by source. Note that we labeled all images as maps or non-maps manually.

Table 1. Distribution of images by source

Source of image (Keyword used)	**Number of images**	**Number of map images**	**Number of non-map images**
Los Angeles Maps	378	327	51
Seattle Maps	132	87	45
Chicago Maps	480	376	104
Pittsburgh Maps	139	92	47
New York Maps	143	87	56
New Delhi Maps	188	124	64
City maps	624	611	13
NonMap (CALTECH 101)	3,082	0	3,082
ALL	*5,166*	*1,704*	*3,462*

Our experimental collection includes not only cities in the United States, but also international cities. Specifically, we included maps from New Dehli, and the maps retrieved for the keywords “city maps” include maps from China, Hungary, Israel, Ireland, France, Scotland and many other countries. Further, the maps in our collection retrieved by city keywords (such as “Pittsburgh maps”) include not only urban street level maps, but also county maps, highway maps, weather maps, “metro area” maps, tourist maps (such as parks), and other types. Therefore, our map collection is diverse and interesting.

[2] images.search.yahoo.com

Our CBIR-based classifier builds upon LIRE[3], an open source CBIR library written in Java,[4] which we augmented to use Water-Filling features[9]. Previous work demonstrated that Water-Filling features increase performance in the retrieval of images with clear edge structures [9], a condition that applies well to maps. Since Water-Filling uses edge maps, we pre-process each image using the Canny edge detector[12] to create the edge maps.

For our experiments, we randomly select 800 maps and 800 non-maps from the entire set of images to build the repository for the CBIR method. Since the repository acts as the set of labeled samples for the CBIR method, we used this same set to train the SVM. Then we test each method on 800 randomly chosen maps and 800 randomly chosen non-maps to test the method's effectiveness. We repeated this process over 10 trials.

Our experiments test two hypotheses. First, we test whether CBIR is a more accurate classifier than the SVM by comparing both methods, using the Water-Filling features. Second, we also test whether Water-Filling features are more suited to map classification by comparing an SVM using Water-Filling features to one that uses Law's Textures [3]. By comparing an SVM using Law's Textures, which was used in a machine learning approach to the same problem [3], to our CBIR-based method using Water-Filling, we can also show that not only is our method more scalable and robust (since it does not require a training phase), it also outperforms the previous approach. Table 2 presents our experimental results as average precision, recall and F_1-measure (the first harmonic mean between precision and recall), the standard metrics for classification tasks.

Table 2. Performance of the CBIR and SVM methods

Method	**Precision**	**Recall**	**F_1-measure**
CBIR with WaterFilling	87.14	77.36	81.96
SVM with WaterFilling	88.80	56.00	68.69
SVM with Laws' Texture	69.50	47.43	56.38

The first result to notice is that the CBIR with WaterFilling method greatly outperforms the other two methods in F_1-measure, which means it has the superior map classification performance. Note that all differences, except for one, are statistically significant, using a two-tailed t-test with α set to 0.05. The only difference that is not statistically significant is the precision between the CBIR with WaterFilling and the SVM with WaterFilling methods.

Note that while the precision using WaterFilling is similar (not statistically significant) using either the SVM or the CBIR method, the recall is much improved using the CBIR method, yielding the best F_1-measure. This supports our notion that CBIR generalizes to many map types better than SVM does. In order to increase the SVM's recall, we would need to train it for the various map classes. As it stands, training it to learn a single "meta-map" class only results

[3] http://www.semanticmetadata.net/lire/
[4] LIRE is part of the Calif&Emir project [11].

in its classification of roughly half of the maps correctly. Meanwhile, because the CBIR method returns the nearest neighbors, which may be of arbitrary map types, it is able to cover many more map classes, and return a higher proportion of the maps. It does this while maintaining a high level of precision, and is therefore a more accurate classifier.

The next result to notice is that WaterFilling features are much better suited for the map classification task. This is clear by examining the difference in precision between the SVM method using WaterFilling features and the SVM method using Law's Textures. Both methods have very similar recall values, generally capturing half of the maps in the testing data, but the Law's Textures method does a far worse job at distinguishing noise from the true maps. As stated, since maps have a strong-edge structure, Water-Filling can do a much better job at distinguishing true maps from false positives.

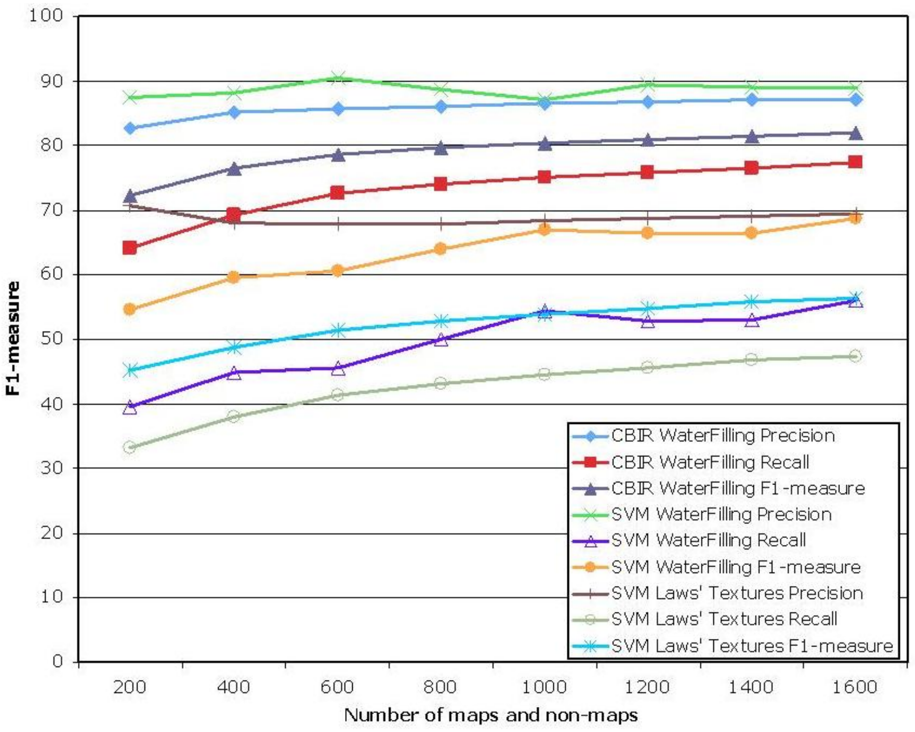


Fig. 5. Comparison of CBIR and SVM varying the repository size

We also analyze the effect of the size of the repository on the CBIR method's classification performance. Since the repository is equivalent to the training data for the SVM method, this is analogous to studying the effect of the amount of training data on the SVM. To do this, we repeated the above classification experiment, this time varying the repository size (training data) by 200 maps and 200 non-maps, starting with 200 maps and 200 non-maps up to 1,400 maps and 1,400 non-maps. Figure 5 again compares the F$_1$-measures for the three methods, CBIR with WaterFilling, SVM with WaterFilling, and SVM with Law's Textures.

Figure 5 suggests that even with a small repository, the CBIR method outperforms the SVM methods. Moreover, as we add more images to the repository, the CBIR method's F$_1$-measure improves steadily. Again, the primary cause for both of the SVM methods' low F$_1$-measure is their low recall. As we argue above, the primary reason for the low recall is that maps vary dramatically in their shapes and density, such that when the SVM tries to learn one model for all of these types as maps, it converges to a set of feature values which are an average over these maps. As a result it defines a hyperplane approximately in the middle of the feature space resulting in lots of maps falling on the non-map side of the plane. In fact the recall hovers around the 50% mark reaching 56% for the most training. Note that the way to overcome this would be train an SVM for each map type, but this is infeasible given that we do not know the map types on the Web ahead on time.

Lastly, we examine the maps misclassified by the CBIR method. Since the precision is high, we focus on issues dealing with recall. Specifically, we delve more deeply into false-negatives (maps that are errantly classified as non-maps), which are the determinant for recall misses. We realized that the majority of false-negative cases are caused when there is one more non-map than map in the returned neighbors, which leads the system to classify the input image as a non-map (since it has a five to four majority). In most of these cases, it is a small set of non-map images that repeatedly get included as these neighbors which sway the vote to an errant classification. We call these "culprit" images. Figure 6 shows the edge-map of an example culprit image. This image is a person in front of a stack of books. However, the edge map for the stack of books in the background looks quite similar to an urban map (it has lots of squares and intersections). To deal with "culprit images" we plan to use relevance feedback techniques to find this small set of non-map images and remove them from the repository.

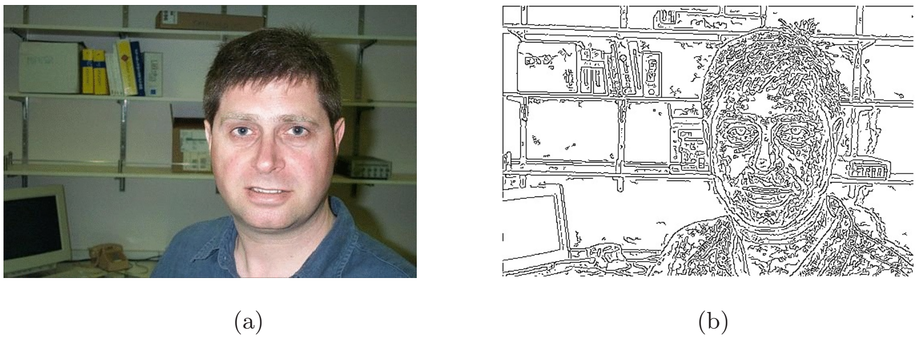

(a) (b)

Fig. 6. An image where the edge map looks like a city map; Figure (a) shows the original image, Figure (b) shows the edge map

4 Related Work

CBIR methods have been applied to various scientific disciplines ranging from astronomy[13] to botany[14]. Much attention has been given to CBIR methods in medicine, where they can have tremendous impact [15]. For example, in one medical system the authors use a CBIR-based, k-Nearest Neighbor approach to classify medical images [16]. This work also includes Water-Filling features for the CBIR component. However, their work (and most of those above) differs from ours in the context in which it is applied. Our system is geared toward automatically harvesting maps from the Web, while their system is used to classify images so that images can be queried categorically. More specifically, these authors use many different types of features to classify their images, while we use only water-filling in the interest of accurate classification. Further, we are the first to use CBIR methods to automatically harvest maps from the Web.

Although we are the first to propose CBIR-based classification for map harvesting, other work has been proposed to automatically classify (and harvest) maps from the Web [3]. As stated in our experiments, Desai, et. al. [3] propose machine learning methods to classify maps versus non-maps. In our experiments we find this method is less effective than our CBIR based classifier. More specifically, not only is the machine learning component not as accurate as the CBIR-based classifier, but a machine learning method requires different models for each map type, since our experiments show that learning a single model to cover all maps leads to inaccurate classification.

Our experiments point to the necessity of using the correct features for CBIR. While we choose Water-Filling, which are good for images such as maps with strong edge maps, other methods could perhaps work as well. For instance, authors have proposed "salient point" features based on wavelets [17]. Another set of features based on shape similarity [18] could be well suited for our task as well, since maps seems to share certain shapes within them. Lastly, methods have been proposed to more efficiently store color information [19], which makes retrieval more efficient. Although we use textures based on edge maps to make our method color invariant, color information might help in discriminating maps. Our CBIR-method is not tied to Water-Filling, though those features perform well. It will be interesting future work to compare the various features and their efficiency and accuracy for map classification.

5 Conclusion

In this paper we present an autonomous, robust and accurate method for classifying maps from images collected on the World Wide Web. We find that a CBIR based classification method outperforms a machine learning based method, largely because we do not have to explicitly model the different types of maps that should be covered by the classifier. That is, by leveraging CBIR we can classify a variety of maps without having to explicitly train a classification model for each one. Further, we find that Water-Filling features, which are shown to work well on images with clear edge structures, work well for maps in our classifier.

However, although our method performs well, there are still areas for improvement. For example, some maps are misclassified when the majority vote is borderline (for example, one image sways the classification as a map or non-map). In this case, we can deal with the ambiguity by employing relevance feedback techniques from information retrieval. Such relevance feedback could help us to identify and prune away “culprit” images who consistently sway the vote in miss-classifications. Other future work involves exploring different features such as wavelets and shape similarities.

Nonetheless, despite the future work proposed above, our technique provides an automatic, accurate, practical and scalable solution to the problem of creating useful image repositories from the Web. More importantly, by plugging our method in with methods for aligning raster maps with satellite images we can create a harvesting framework for scouring the freely available images on the Web to build map collections for any given region in the world.

Acknowledgment

The authors thank Vipul Verma for helping implement the Water-Filling features and CBIR method.

References

1. Chen, C.C., Knoblock, C.A., Shahabi, C.: Automatically conating road vector data with orthoimagery. Geoinformatica 10(4), 495–530 (2006)
2. Chen, C.C., Knoblock, C.A., Shahabi, C.: Automatically and accurately conating raster maps with orthoimagery. GeoInformatica (in press, 2008)
3. Desai, S., Knoblock, C.A., Chiang, Y.Y., Desai, K., Chen, C.C.: Automatically identifying and georeferencing street maps on the web. In: Proceedings of the 2nd International Workshop on Geographic Information Retrieval (2005)
4. Chiang, Y.Y., Knoblock, C.A., Shahabi, C., Chen, C.C.: Accurate and automatic extraction of road intersections from raster maps. Geoinformatica (in press, 2008)
5. Chiang, Y.Y., Knoblock, C.A.: Classification of line and character pixels on raster maps using discrete cosine transformation coefficients and support vector machines. In: Proceedings of the 18th International Conference on Pattern Recognition (2006)
6. Chiang, Y.Y., Knoblock, C.A., Chen, C.C.: Automatic extraction of road intersections from raster maps. In: Proceedings of the 13th ACM International Symposium on Advances in Geographic Information Systems (2005)
7. Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 1349–1380 (2000)
8. Fix, E., Hodges, J.L.: Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical report 4. USAF School of Aviation Medicine, Randolph Field, TX (1951)
9. Zhou, X.S., Rui, Y., Huang, T.S.: Water- lling: A novel way for image structural feature extraction. In: Proceedings of the International Conference on Image Processing, pp. 570–574 (1999)

10. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In: Proceedings of IEEE CVPR Workshop on Generative-Model Based Vision (2004)
11. Lux, M., Becker, J., Krottmaier, H.: Caliph&emir: Semantic annotation and retrieval in personal digital photo libraries. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, Springer, Heidelberg (2003)
12. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Analysis and Machine Intelligence 8, 679–714 (1986)
13. Csillaghy, A., Hinterberger, H., Benz, A.O.: Content based image retrieval in astronomy. Information Retrieval 3(3), 229–241 (2000)
14. Wang, Z., Chi, Z., Feng, D.: Fuzzy integral for leaf image retrieval. In: Proc. of IEEE Intl. Conference on Fuzzy Systems (2002)
15. Müller, H., Michoux, N., Bandon, D., Geissbuhler, A.: A review of content-based image retrieval systems in medical applicationsclinical benefits and future directions. International Journal of Medical Informatics 73, 1–23 (2004)
16. Lehmann, T.M., Güld, M.O., Deselaers, T., Keysers, D., Schubert, H., Spitzer, K., Ney, H., Wein, B.B.: Automatic categorization of medical images for content-based retrieval and data mining. Computerized Medical Imaging and Graphics 29, 143–155 (2005)
17. Tian, Q., Sebe, N., Lew, M.S., Loupias, E., Huang, T.S.: Image retrieval using wavelet-based salient points. Journal of Electronic Imaging 10(4), 835–849 (2001)
18. Latecki, L.J., Lakamper, R.: Shape similarity measure based on correspondence of visual parts. IEEE Trans. Pattern Analysis and Machine Intelligence 22(10), 1185–1190 (2000)
19. Deng, Y., Manjunath, B.S., Kenney, C., Moore, M.S., Shin, H.: An efficient color representation for image retrieval. IEEE Trans. Image Processing 10(1), 140–147 (2001)

Improving Localization in Geosensor Networks through Use of Sensor Measurement Data

Frank Reichenbach[1], Alexander Born[2], Edward Nash[2], Christoph Strehlow[1], Dirk Timmermann[1], and Ralf Bill[2]

[1] University of Rostock, Germany
Institute of Applied Microelectronics and Computer Engineering
{frank.reichenbach,christoph.strehlow, dirk.timmermann}@uni-rostock.de
[2] University of Rostock, Germany
Institute for Management of Rural Areas
{alexander.born,edward.nash,ralf.bill}@uni-rostock.de

Abstract. The determination of a precise position in geosensor networks requires the use of measurements which are inherently inaccurate while minimizing the required computations. The imprecise positions produced using these inaccurate measurements mean that available methods for measurement of distances or angles are unsuitable for use in most applications. In this paper we present a new approach, the Anomaly Correction in Localization (ACL) algorithm, whereby classical trilateration is combined with the measurements of physical parameters at the sensor nodes to improve the precision of the localization.

Simulation results show that for localization using triangulation of distance measurements with a standard deviation of 10% then the improvement in precision of the estimated location when using ACL is up to 30%. For a standard deviation in the measurements of 5% then an improvement in positioning precision of ca. 12% was achieved.

1 Introduction

In the near future masses of tiny smart electronic devices will be placed ubiquitously in the environment surrounding us. These so called geosensor networks (GSN) sustainably measure mechanical, chemical or biological conditions, aggregate the important information and transmit it over neighboring sensor nodes to a data sink where it can be collected and then analyzed. GSNs are evolving as a promising technique for industry, modern life science applications or natural disaster warning systems. One of the most important issues in GSNs is the localization of each node using distances or angles to neighbors as the geographical position is required to make use of the measurements. Due to the large errors which are caused by available methods for the measurement of distances or angles, the expected precision, particularly indoors, is insufficient for most applications.

In our new approach we take advantage of the fact that sensor measurements are related to their position in a pattern that can be largely determined in advance. We have therefore developed a model that combines spatial sensor information and classical localization approaches such as trilateration to increase the overall precision.

This paper is structured as follows. In Section 2 the fundamentals of localization in geosensor networks are summarized. Section 3 introduces the basic concept by which we hope to improve the precision of localization and Section 4 describes the new ACL algorithm in detail. Section 5 then presents simulation results before Section 6 summarizes the results presented in this paper. Finally, Section 7 discusses some ideas for future work in this area.

2 Fundamentals of Localization

The task of the described networks consists of the collection with sensors of a phenomena with a certain spatial dimension. The capability to self-determine the position of each sensor is an essential feature of the sensors because measurements are only useful if connected to a time and a place. A potential method would be the use of the Global Positioning System (GPS) or in future Galileo [1]. Because of the additional costs and the required small size of the sensor nodes these techniques are only feasible if used on a small number of more powerful nodes, further called Beacons. These Beacons determine their positions with such methods and make this information available to all other sensor nodes in the network. For the localization of the other nodes, different methods are available and can be separated into approximate and exact methods. An overview can be found in [2],[3].

Simple exact methods are the well known trilateration or triangulation, using measurement of distances or angles respectively. Due to the fact that measuring angles requires additional hardware (e.g. antenna arrays on each node), triangulation has not been subject to intensive investigation and the trilateration approach is favored. In the case of 2D localization, a system of three equations may be constructed using Euclidian distances [4]. Subtracting one equation from the other two and insertion of one of the remaining unknowns produces a quadratic equation which may be uniquely resolved. This method requires a low calculation and memory effort, but the measurement of the distances results in systematic and stochastical errors. These errors influence the results of the trilateration and significantly decrease the accuracy of the determined position because of the lack of robustness of the trilateration. In the worst case the localization process can fail if the Beacon positions are also subject to errors.

Two methods to overcome these problems exist in the literature. Approximative algorithms use coarse positions or distances as start values and determine relatively inaccurate results, but with a minimum of calculation cost. Algorithms used include e.g. (weighted) centroid determination (CL,WCL) [5,6] or overlapping areas (APIT) [7] and are more error resistant and therefore robust, but the approximate nature of the algorithms themselves means that even with error-free start values an exact position can never be determined.

This is an advantage of the second class of methods - the exact algorithms. These can produce exact positions if accurate start values are used. The disadvantage is that they require a higher calculation and memory effort. They are therefore not appropriate to be used at resource-limited sensor nodes. There are however some algorithms which overcome these problems by distributing the tasks [8,9,10,11].

Additional to further developments of the algorithms we follow up a completely different approach - the use of spatial information inherent in sensor measurements. We will discuss this concept in the next section.

3 Basic Concept

In Section 2 we explained the problems affecting the positioning process. Intensive efforts have been made to reduce these errors using better localization algorithms. However, these algorithms all base on the same data such as Beacon positions and distance measurements. Even with the best measurement methods some error sources cannot be eliminated. Further available sources of information must therefore be considered. With the large amount of sensor data collected it is possible to draw conclusions about the environment in which the sensor nodes are located. In many cases there will be a correlation between the collected data and the sensor location. This information can be used to increase the position accuracy if it is possible to mathematically define this correlation.

For a better understanding we will give an application example from Precision Agriculture, where the use of GSNs is currently a topic of research [12],[13]. In the future a large number of sensor nodes may be deployed, e.g. by aeroplane, over cropland. The sensor nodes measure chemical or biological soil composition as well as typical physical parameters such as temperature, light intensity or air pressure. Nodes which settle in areas of dense plant canopy, e.g. under trees, will measure a consistently lower light intensity, caused by the shadowing effects of the vegetation. On the other hand the sensor nodes in an open area without plant coverage would measure a consistently higher light intensity. Combining the estimated position with the sensor data and a surface model would allow us to draw conclusions about the localization. If a sensor node estimates a position in an open area but records a very low light intensity then an outlier is highly probable. Although this example is highly specialized, due to the fact that the sensor nodes may carry a large number of different sensors many potential classes of measurements could be used for an outlier detection.

In this paper this principle is used as follows. Usually a sensor network contains redundant Beacons. Using trilateration, different positions can be estimated followed by the elimination of errors. In Section 4 we will introduce an algorithm with which it is possible to filter values which are subject to errors and thus improve the localization result. This approach is based on the idea that it is possible to define location-dependent ranges for measurement of certain phenomena. Using these ranges the sensor nodes can exclude certain areas for determining their own position, leading to a decrease in the localization error.

4 Anomaly Correction in Localization (ACL)

In this section we describe the Anomaly Correction in Localization (ACL) method. The aim can be summarized as to produce a very resource-saving trilateration, requiring only minimal additional calculation on the nodes whilst still determining a precise position. In particular, single outliers from the distance measurement may significantly affect the

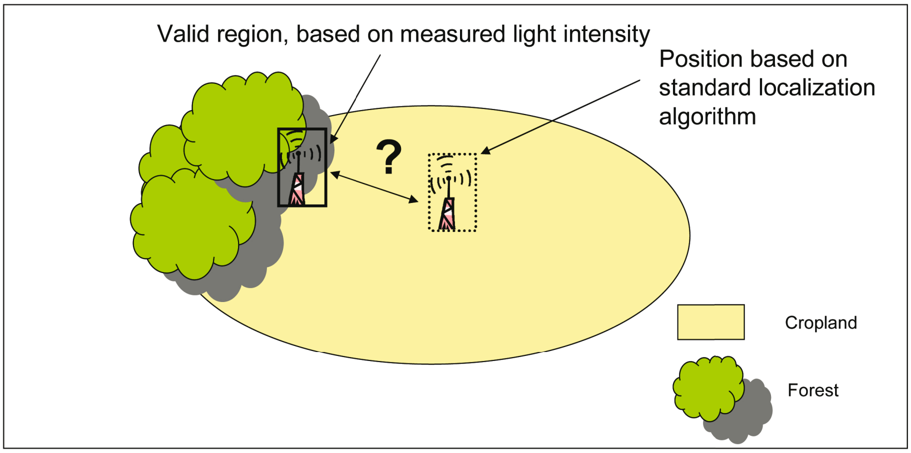

Valid region, based on measured light intensity
Position based on standard localization algorithm
?
Cropland
Forest

Fig. 1. Resulting discrepancy in position between sensor measurement and calculated trilateration

result of the trilateration. This effect should be reduced or removed by the use of the ACL algorithm as described here.

4.1 Prerequisites

ACL realizes an improvement in the localization through elimination of highly inaccurate estimated positions based on sensor measurements and some previous knowledge of the measurement environment. Figure 1 illustrates how a sensor node with light sensor may reject false positions when the light conditions for cropland and forest are known. In this case, based on the measured light conditions, only positions in the forested area will be accepted. Prerequisites for a successful application of ACL are:

– there must be sufficient sensors installed on the sensor nodes which measure a spatially-related phenomenon (exact numbers and densities are likely to be application dependent and further investigation is required),
– and it must be possible to clearly define the spatial relation of the phenomenon being used using discrete (or potentially, with an extension of the model, fuzzy) zones linked to expected observation values which can be determined independent of the current sensor network (e.g. zones for expected soil moisture content may be derived from a digital terrain model through use of the topographic wetness index).

Additionally it must be possible to allocate a specific region of space to each environmental parameter. This important relationship between a defined spatial region and a sensor value range is hereafter referred to simply as 'sensor interval'. Potential environmental parameters which may be used are temperature, light intensity, moisture, pH, pressure, sound intensity and all other physical parameters which may be metrically defined. A sensor interval is then defined as matching to a region when the expected measurement value lies within the delimited range with a high degree of probability.

A further important prerequisite is an inhomogeneity in the relevant environmental parameter. If the entire measurement field belongs to a single sensor interval or the measured variable may not be classified then this parameter is not suitable for use in the localization. It is therefore only possible to use parameters which may be divided into two or more intervals with a high significance.

The ACL algorithm bases on the previous trilateration. Since normally more than the three required Beacons (n Beacons), are available within the range of a sensor node, it is possible to perform multiple trilaterations. In total, $\binom{n}{3}$ different trilaterations may be calculated. The question however remains as to which trilateration will produce the best (most precise) result, as it is important to minimize the computational effort. This optimum may be expressed by the question as to which trilateration produces a result which does not violate the clear rules of the ACL algorithm. Based on our previous example, this means that there would be a conflict when an estimated position lies in an open area, but the light sensor measures a very low value. This indicates that this trilateration is unreliable. The mathematical background is described in the following Section.

4.2 Construction of the Model

A model has been developed which abstracts real given environmental parameters and allows a variable allocation to intervals. For this, a rectangular measurement area is defined in which the sensor nodes are placed. The whole area is subdivided using an arbitrarily scalable raster, which may be represented using a square matrix. This allows both a very flexible configuration and a fast computation.

Sensor intervals do not have preassigned values but may be indicated by a region in which the sensor values cluster within a certain interval. These regions may be formed by a set of adjacent cells, whereby each cell is represented by a logical ”1” in the relevant position in the matrix. This significantly simplifies the required calculations as no geometrical tests (e.g. point-in-polygon) must be performed but only a simple mapping of the spatial coordinates to the dimensions of the matrix and a logical comparison. The size, number and position of the sensor intervals may be generically defined.

The measurement of sensor values is modeled through the allocation of the nodes to the sensor intervals in which the actual position of the sensor lies. A potential measurement error is thereby excluded, i.e. each sensor delivers values from the exact interval corresponding to its position. The measured sensor information are further referred to as ’sensor profile’ and are considered to be a defined spatially-variable property.

Figure 2 illustrates an example measurement area. Each node has a 1D vector which defines its allocation to each sensor interval. Node 1 (N_1) has in this case the vector $V(N_1) = (1, 0, 0)$ which indicates that N_1 is located in sensor interval 1, but outside intervals 2 and 3.

It is also possible that a sensor interval is distributed over multiple spatially discrete regions in order to allocate the same sensor profile to multiple areas. Such discrete regions may generally be considered as separate sensor intervals, with the difference that calculated positions which lie in one sub-region, but where the actual position lies in a different sub-region, will not be recognized as outliers.

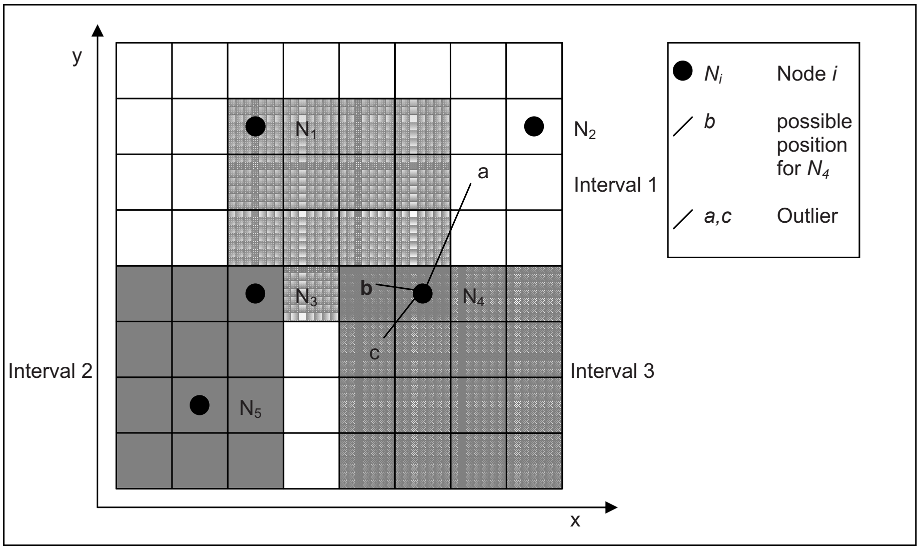


Fig. 2. Schematic of a localization with outlier detection by ACL

4.3 Algorithm Description

There now follows a step-by-step description of the algorithm. There are two possible sequences which should be considered.

ACL with Averaging. The following sequence is more computationally expensive, but potentially more accurate as all available information is used.

1. Beacons send their position to the sensor nodes in the transmission range t_x.
2. Sensor nodes save the received position and measure the distance to the Beacons using a standard measurement technique such as signal-strength measurement.
3. After all data are saved, i.e. all Beacons have finished transmitting, the sensor nodes calculate all $q = \binom{n}{3}$ possible positions by repeated trilateration.
4. All resulting positions are tested using the ACL algorithm. The actual measured sensor value is compared to the previously defined sensor interval. If it matches then the position is considered as valid, otherwise as invalid.
5. After all positions have been tested using ACL, an average of all valid positions is calculated. This average is then used as the final estimated position.

ACL with First Valid Position. As the previously described algorithm has the disadvantage of requiring that all trilaterations are calculated, we introduce a version which may be interesting for practical use. In this version, the calculation is stopped as soon as a valid trilateration is found. The sequence is identical to that previously described, except that the result of each trilateration (stage 3) is immediately validated using ACL (stage 4) until a valid position is found. This position will then be used as the estimated position of the node and the algorithm terminates. Using this version then a greater localization error is to be expected. This is explained in the following Section.

5 Simulation and Results

For simulation we have used the packet simulator J-Sim from the Ohio State University [14]. Details of the implementation can be found in [15]. At this point we wish to present the simulation conditions and the results.

The necessary distance measurement for the trilateration is modeled through calculation of the Euclidean distance, adjusted using a normally-distributed (Gaussian) random value to represent measurement error. The simulated distance d is thus calculated using the following formula:

$$d = \sqrt{(x - x_0)^2 + (y - y_0)^2} + r_{Gauss} \cdot \sigma_d \quad (1)$$

where x, y are the coordinates of the unknown node, x_0, y_0 are the coordinates of the Beacons, r_{Gauss} is a normally distributed random number between zero and one and σ_d is the standard deviation of distances.

For the localization an even distribution of nodes is assumed, resulting in an equal distribution of favorable and unfavorable alignments for the trilateration. This means that even with this idealized normally-distributed error model then significant outliers in the localization are to be expected.

In the simulations then the localization errors of the initial position E_f and the localization error of the first correct position (without outliers) $E_{f,ACL}$ are determined. It can therefore be seen, by how much the mean localization error is reduced when the ACL algorithm with version 1 and 2 is applied. The improvement is shown in the diagrams in meters, although since a sensor field of 100m $\times$ 100m is used then this may also be interpreted as percent. Similarly, the standard deviation may also be interpreted as percent values. A 100m $\times$ 100m sensor field with a cell size of 5m $\times$ 5m was used for all simulations. Figure 3 shows a typical visualization window of the results achieved by the J-Sim tool.

5.1 Mean Improvement with 10 Beacons

In the first simulation, 10 Beacons and 20 sensor nodes are distributed in the field. Three sensor intervals with range 49 and a distance error of $\sigma_d = 1.0$ are used. In the case that all three sensor intervals do not overlap, their area represents ca. 36% of the total area. In a simulation with 10 Beacons there are 120 possible trilaterations. At this point, only the mean improvement $\overline{V} = \overline{E} - \overline{E}_{ACL}$ is considered. The data are sorted based on the proportion of outliers, i.e. it is calculated how many of the 120 possible positions lie outside the correct sensor interval.

For each of the 20 nodes used there is a particular percentage of outliers and therefore a line in the diagram (figure 4). Each line represents the absolute value of the mean improvement which each node achieves with 120 trilaterations and the given number of outliers. In the case that multiple nodes have the same number of outliers a mean value is calculated.

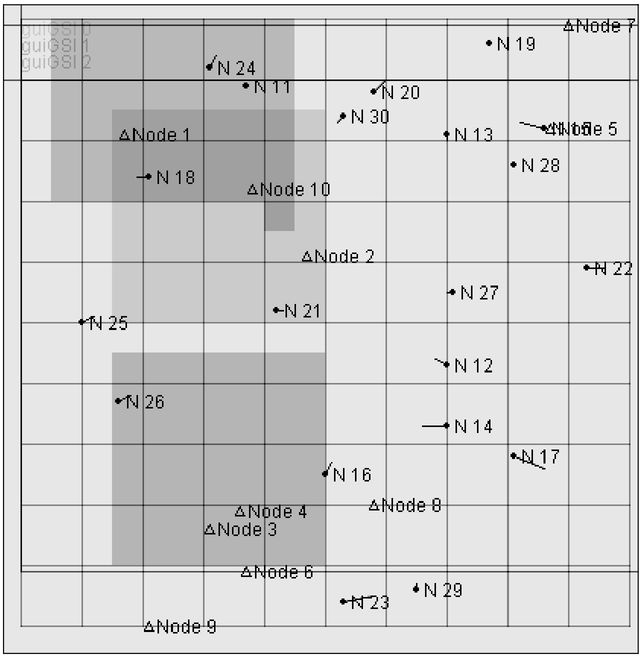


Fig. 3. Screenshot of simulation results in J-Sim; triangles are Beacons and points with lines are sensor nodes with its error-vector; three darker areas represent sensor intervals

Since the raw data show a high scattering, a linear regression is used to emphasize the trend. The improvement increases with an increasing number of outliers. This trend is not demonstratably linear, but a similar trend is observed over many simulations, leading to the assumption that the improvement is at least linear or even exponential. Similarly it can be seen that with a low number of outliers, up to ca. 15%, there is a negative improvement (i.e. a worsening). This shows that there is a certain minimum number of outliers under which the use of the ACL algorithm is not appropriate. This limit is dependent on the particular configuration and can not therefore be generally specified.

5.2 Correlation between Number of Outliers and Distance Errors

The number of outliers is dependent on the error in the distance measurement, in this case the standard deviation σ_d. In the following simulation the same configuration as in Section 5.1 is used with different distance errors. The parameter σ_d is varied between 1.0 and 10.0 with an interval of 1.0. The number of outliers is calculated as a function of this parameter.

Figure 5 shows the mean number of outliers in relation to σ_d over multiple simulations. It shows that the number of outliers increases with an increasing standard deviation. Again a trend line is shown to illustrate the trend. This emphasizes that the values have a high random component and deliver different results in each simulation. It is however clear that the mean proportion of 15% outliers is rapidly exceeded, which

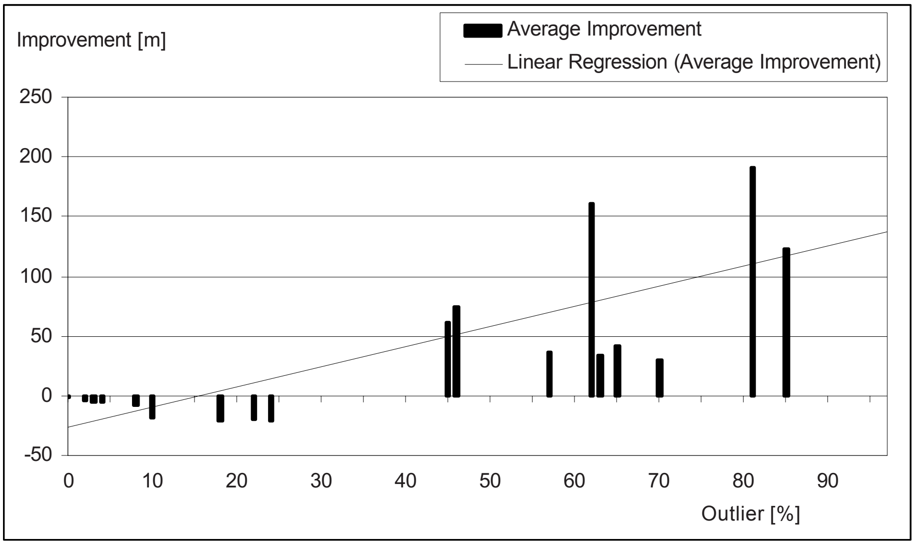


Fig. 4. Average improvement over outliers with 10 Beacons, 20 sensor nodes and a standard deviation $\sigma_d = 1.0$

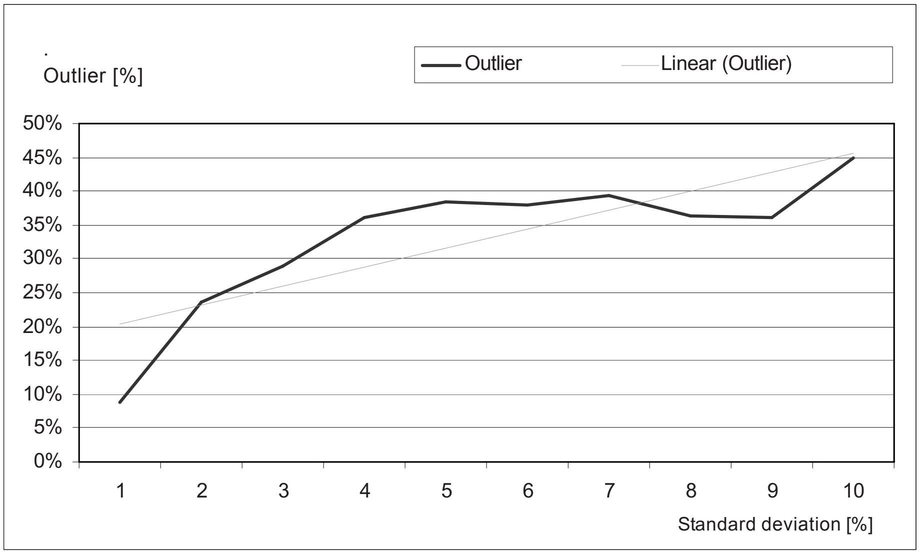


Fig. 5. Average outliers depending on the standard deviation σ_d with 10 Beacons and 20 sensor nodes

indicates that the ACL algorithm will only be ineffective, or detrimental as in figure 4, where there is a very small σ_d.

5.3 Influence of the Size and Number of Sensor Intervals

Further important factors in the effectiveness of ACL are the size, number and position of the sensor intervals. For the simulations, the intervals were individually parameterized by size and number, but randomly positioned. Alongside the random distribution of the sensors, this is one of the main reasons for the high variability of the resulting data. There is therefore no explicit results from which a formula for the accuracy of the ACL in relation to these parameters can be derived. An investigation of the influence of size and number of sensor intervals however resulted in the expected results. An increasing number of sensor intervals results in an increasing fragmentation of the measurement field into smaller sections with different sensor profiles. This is particularly the case where the sensor intervals are larger and therefore overlap more. The number of outliers thus increases with an increasing size and number of sensor intervals, which results in a greater improvement. It is also important to note that sensors which lie near to an interval boundary profit most from the outlier detection because the probability of an outlier is higher in these locations than in a homogenous region.

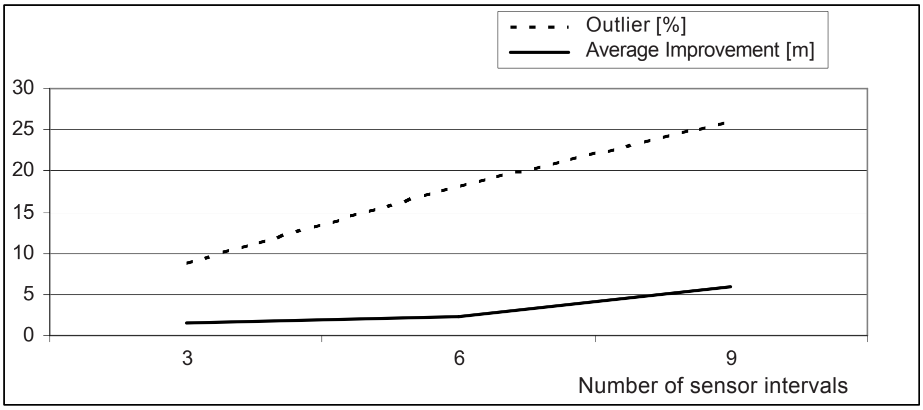


Fig. 6. Outlier and average improvement depending on the number of sensor intervals with 10 Beacons and 20 sensor nodes

Figure 6 shows this relationship. With an increasing number of outliers, the mean improvement $\overline{V}$ also increases with an increasing number of sensor intervals. For this case a sensor interval with three sub-regions of size 49 was created. In a second step a further similar sensor interval was introduced and subsequently a third, whereby the previous intervals were retained with a constant configuration. Furthermore, 10 Beacons, 20 nodes and $\sigma_d = 1.0$ were used, which indicates that with increasing distance errors both curves would show an increased slope.

5.4 Comparison of Mean Improvement and First Improvement

Due to the various possibilities for determining a final estimated position, experiments were made which should favor each method. For this a field with three sensor intervals of size 49 with 10 Beacons and 20 nodes was used to compare the mean improvement $\overline{V}$ with the first improvement V_f. The mean values from the 20 nodes were used to reduce the weight of the variation. The standard deviation of the distances was again varied between 1.0 and 10.0.

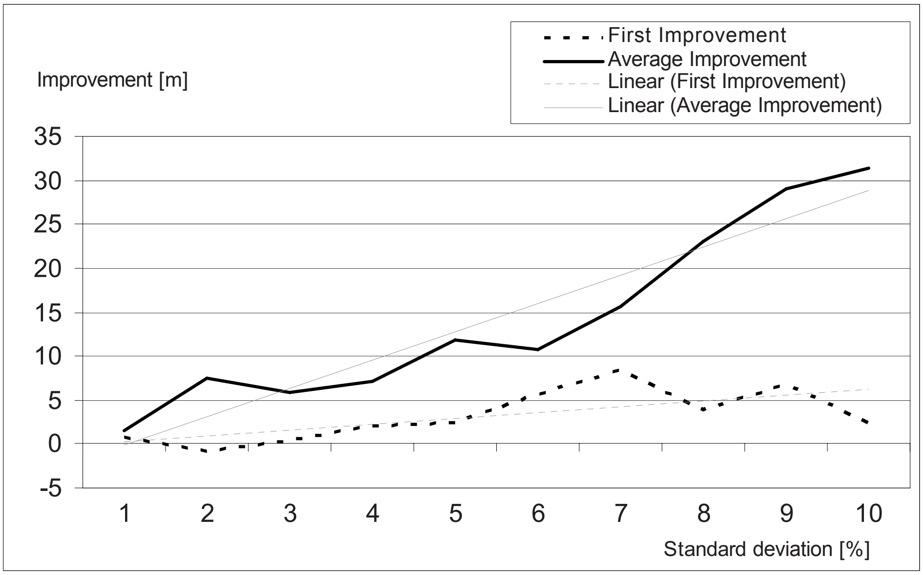


Fig. 7. Comparison of the average improvement and the first improvement depending on the standard deviation σ_d

It is clear from figure 7 that the first improvement demonstrates a significantly lower increase in precision in comparison with the mean improvement. This is partly due to the fact that significantly fewer trilaterations are considered, but also because only a few nodes show any first improvement because they are not near the boundary of an interval and therefore have few or no outliers. The trends are also in this case not necessarily linear and are emphasized using regression lines. The mean improvement is demonstrated here to deliver significantly better results and is therefore the method of choice, although the increased computational expense must also be considered.

6 Conclusion

This paper presented a new approach to improve the precision of localization algorithms in sensor networks which were previously too inexact or calculation-intensive for use. The spatial correlation of the measured sensor values is used for this. The principle used is that sensor observations vary within defined boundaries depending on their location. We have presented a way to use this information by defining sensor intervals. These may be used after localization by trilateration to test the reliability of the resulting estimated positions by comparing the observed values with the sensor intervals. This method was presented as the ACL algorithm together with investigations as to its effectiveness.

Simulations showed that the use of the ACL algorithm can improve the mean error of the localization using trilateration by up to 30% when the measured distances have a standard deviation of 10%. With a 5% standard deviation the improvement is ca. 12%. These results were obtained using three known sensor intervals, which coverered around one third of the measurement field. With further prerequisites the localization could be further improved. Only with very small distance errors with a standard deviation of under 1% is the use of the algorithm not appropriate as in this range an increased localization error was determined.

In conclusion, it has been shown that using the sensor measurement data for localization is a sensible approach, particularly as these data are already available. A significant improvement is possible with minimal additional complexity of calculation on the sensor nodes. The concrete use is however very dependent on how accurate the available measurements are and how many prerequisites can be determined. An advantage of this method is that no additional errors are produced since it effectively involves no more than a filtering of the input values. This can have negative consequences in isolated cases, but from an overall standpoint leads to a more precise result. A further advantage of the model presented here is the possibility to formulate arbitrarily complex prerequisites.

The ACL algorithm can be considered as an efficient additional method for localization. Even with relatively few preconditions it is possible to improve the localization or to verify the positions. In particular in combination with approximate algorithms it is possible to obtain good results, but also with exact positioning where large distance errors are present then the ACL is of benefit.

7 Future Work

The idea presented here is by no means optimally applied. It is necessary to find a better mathematical model on which the ACL algorithm may operate. Currently a simple comparison of measured values with sensor values is made. This is computationally easy, but the spatial information can only be very coarsely exploited. A model such as convex optimization should be considered instead.

Furthermore, the definition of sensor intervals is still very static and inflexible. This could also lead to problems at run-time as the sensor intervals may vary over time. To continue the example of light intensity sensors, the intervals may vary strongly between day and night or summer and winter. To solve this problem an automatic interval generation may be used, which may be possible with e.g. an evolutionary algorithm or simmulated annealing.

Ultimately the additional information are used purely as a means of detecting outliers in the trilateration. The aim is to combine further localization algorithms with this information to obtain an increased precision and to possibly weight the defective distance measurements and error prone sensor data. It is conceivable that the briefly introduced WCL algorithm may be thus made more precise. This is however primarily reliant on the effectiveness of new mathematical models.

Acknowledgment

This work was supported by the German Research Foundation under grant number TI254/15-1 and BI467/17-1 (keyword: Geosens). Thanks also to the reviewers who took the time to provide constructive criticism which has helped improve this paper.

References

1. Gibson, J.: The Mobile Communications Handbook. CRC Press, Boca Raton (1996)
2. Reichenbach, F.: Resource-aware Algorithms for exact Localization in Wireless Sensor Networks. PhD thesis, University of Rostock (2007)
3. Stefanidis, A., Nittel, S.: GeoSensor Networks. CRC Press, Boca Raton (2004)
4. Bulusu, N., Heidemann, J., Estrin, D.: Adaptive beacon placement. In: 21st International Conference on Distributed Computing Systems, pp. 489–498 (2001)
5. Bulusu, N.: Gps-less low cost outdoor localization for very small devices. IEEE Personal Communications Magazine 7, 28–34 (2000)
6. Blumenthal, J., Reichenbach, F., Timmermann, D.: Precise positioning with a low complexity algorithm in ad hoc wireless sensor networks. PIK - Praxis der Informationsverarbeitung und Kommunikation 28, 80–85 (2005)
7. He, T., Huang, C., Blum, B.M., Stankovic, J.A., Abdelzaher, T.: Range-free localization schemes for large scale sensor networks. In: 9th Annual International Conference on Mobile Computing and Networking, San Diego, CA, USA, pp. 81–95 (2003)
8. Reichenbach, F., Born, A., Timmermann, D., Bill, R.: Splitting the linear least squares problem for precise localization in geosensor networks. In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 321–337. Springer, Heidelberg (2006)
9. Reichenbach, F., Born, A., Timmermann, D., Bill, R.: A distributed linear least squares method for precise localization with low complexity in wireless sensor networks. In: 2nd IEEE International Conference on Distributed Computing in Sensor Systems, San Francisco, USA, pp. 514–528 (2006)
10. Savvides, A., Han, C.C., Srivastava, M.B.: Dynamic fine grained localization in ad-hoc networks of sensors. In: 7th ACM MobiCom, Rome, Italy, pp. 166–179 (2001)
11. Doherty, L., Pister, K.S.J., Ghaoui, L.: Convex position estimation in wireless sensor networks. In: 20th Annual Joint Conference of the IEEE Computer and Communications, Anchorage, AK, USA, pp. 1655–1663 (2001)
12. Vellidis, G., Garrick, V., Pocknee, S., Perry, C., Kvien, C., Tucker, M.: How wireless will change agriculture. In: Stafford, J.V. (ed.) Precision Agriculture 2007, pp. 57–67 (2007)
13. Konstantinos, K., Panagiotis, K., Apostolos, X., George, S.: Proposing a method of network topology optimization in wireless sensor in precision agriculture. In: Proceedings of the 6th European Conference on Precision Agriculture (2007)
14. J-sim: A component-based, compositional simulation environment (2007), http://www.j-sim.org/
15. Strehlow, C.: Genauigkeitserhoehung der Lokalisierung in drahtlosen Sensornetzwerken durch Einbeziehung von Sensormessdaten, Seminar Paper (2007)

Simplest Instructions: Finding Easy-to-Describe Routes for Navigation

Kai-Florian Richter[1] and Matt Duckham[2]

[1] Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition
Universität Bremen, Germany
richter@sfbtr8.uni-bremen.de

[2] Department of Geomatics, The University of Melbourne, Australia
mduckham@unimelb.edu.au

Abstract. Current applications for wayfinding and navigation assistance usually calculate the route to a destination based on the shortest or fastest path from the origin. However, numerous findings in cognitive science show that the ease of use and communication of route instructions depends on factors other than just the length of a route, such as the number and complexity of decision points. Building on previous work to improve the automatic generation of route instructions, this paper presents an algorithm for finding routes associated with the "simplest" instructions, taking into account fundamental principles of human direction giving, namely decision point complexity, references to landmarks, and spatial chunking. The algorithm presented can be computed in the same order of time complexity as Dijkstra's shortest path algorithm, $O(n^2)$. Empirical evaluation demonstrates that the algorithm's performance is comparable to previous work on "simplest paths," with an average increase of path length of about 10% compared to the shortest path. However, the instructions generated are on average 50% shorter than those for shortest or simplest paths. The conclusions argue that the compactness of the descriptions, in combination with the incorporation of the basic cognitive principles of chunking and landmarks, provides evidence that these instructions are easier to understand.

1 Introduction

Automated wayfinding assistance is an increasingly popular and economically important application area for geographic information science. Systems for automated wayfinding assistance employ computationally efficient algorithms for calculating the shortest or fastest route to a destination, but typically do not account for human principles of direction giving, and ignore how humans conceptualize their environment. The instructions generated, while generally usable, seem artificial and often make it hard to form survey knowledge about imminent wayfinding decisions, i.e., to prepare for what is coming up.

In recent years several approaches have emerged that cover (at least part of) the generation of route instructions that respect for human principles of wayfinding and direction giving [1,2,3,4]. However, these approaches usually aim at improving the presentation of route instructions for a previously calculated route. This paper presents a new algorithm that addresses the problem of *finding* the best route with respect to the simplicity of *route instructions*.

Given a geographic network, our algorithm as explained in this paper finds the route between a source and destination that is the "simplest" to describe, in terms of the complexity of its associated routing instructions. Building on fundamental principles of human direction giving, the route instructions generated are expected to be easier for a human wayfinder to remember, communicate, and use. The underlying algorithm is based on the widely-known Dijkstra's shortest path algorithm [5]. As will be demonstrated in the paper, the extensions made to Dijkstra's algorithm do not increase computational complexity. Thus, the primary contribution of this paper is an efficient algorithm for generating cognitively ergonomic route instructions.

The next section presents related work on automatic generation of cognitively ergonomic route instructions, in particular *simplest paths* and *context-specific route instructions*. Section 3 introduces the algorithm that allows for finding the best route instructions to guide a wayfinder from origin to destination, including a discussion of the computational and cognitive properties of the algorithm. Section 4 then presents the results of an empirical evaluation of the algorithm, looking at the length of the paths and the instruction sequences generated. Section 5 concludes the paper with a discussion of the paper's contribution and an outlook on future work.

2 Generating Route Directions

This paper deals with determining a route between two points in a network space. Efficient algorithms exist for this task, primarily Dijkstra's shortest path algorithm [5]. Shortest path algorithms apply a cost function that is somehow related to the network's structure in its embedding geographical reference frame (e.g., distance between vertices, speed of movement, or direction of travel). However, shortest path algorithms neither account for human conceptualization of space nor for principles of human direction giving.

Human route instructions reflect the instruction-giver's knowledge about an environment. When asked to provide instructions, humans activate the spatial knowledge of the route to be described, identify the relevant information, structure this information, and communicate it to the requester [6,7]. The literature specifically includes two important principles that are employed when providing instructions: 1) references to landmarks and 2) combining multiple consecutive decision points into a single instruction, termed *spatial chunking* by Klippel et al. [8]. Landmarks are important for acquiring and organizing knowledge about our surrounding space [9,10]. In route instructions, they are frequently referred to; landmarks may signal crucial actions, locate other landmarks in relation to the referenced landmark, or confirm that the right track is still being followed [7,11]. Often, humans subsume instructions for several decision points into a single 'chunked' one. For example, "turn right at the third intersection" corresponds to going straight at the next two decision points and then turning right at the third one.

A range of existing research has addressed the automatic generation of route instructions that account for human principles of direction giving. Some of this work only covers parts of the generation process, such as the identification [12,13] or integration [14,15] of landmarks. Others focus on generating instructions that mimic the way humans present such information [1,3], or that adapt to human conceptualization of wayfinding situations [4].

In the following, two approaches are presented in more detail that form the basis for the new algorithm proposed in this paper. The first approach is that of *simplest paths* [16], which aims to find a route that is easy to follow, by minimizing the number and complexity of decision points. The second approach is that of *context-specific route instructions* [17], which aims to generate route instructions for a *given* route that are easy to conceptualize and to remember.

2.1 Simplest Paths

Duckham and Kulik [16] extend standard shortest path search by a heuristic that associates a cost with each pair of connected edges (rather than each edge as in classic shortest path approaches). This cost reflects the complexity of negotiating the “decision point” represented by the two adjacent edges (e.g., turning from one edge onto another — see Figure 1). The specific weighting used is based on an adaptation of earlier work by Mark [18], who classifies different types of intersections according to the complexity of describing the action to be performed there. Duckham and Kulik, accordingly, term their algorithm *simplest path* algorithm. In a simulation experiment, they show that their algorithm generally results in paths that are only slightly longer than the shortest path.

While the costs employed account for structural differences of intersections, they do not account for functional aspects, for example, (possible) ambiguity in the direction to take at an intersection, nor landmarks or other environmental characteristics that might be exploited in instructions. In summary, like shortest paths, the simplest path finds the cheapest route according to a cost function. Unlike shortest paths, the cost function used applies to the complexity of navigation decisions rather than travel distance or time.

2.2 Context-Specific Route Directions

Context-specific route directions account for environmental characteristics and a route’s properties; they adapt to the action to be taken in the current surrounding environment. Such instructions are termed “context-specific” because of the explicit adaptation to the structure and function in wayfinding [17]. A computational process, called GUARD (Generation of Unambiguous, Adapted Route Directions), has been developed by Richter [19] for generating context-specific route instructions. GUARD unambiguously describes a specific route to the destination, with instructions adapted to environmental characteristics. The route that is described has been previously selected outside of GUARD. Figure 2 provides an overview of the generation process.

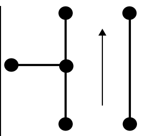
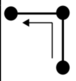
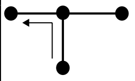
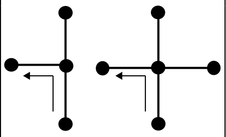
straight on		1 slot
turn (not at intersection)		4 slots
turn left or right at T-intersection		6 slots
turn left or right at intersection		$5 + deg(v)$ slots

Fig. 1. Weighting (slot values) of different intersection types; from [16] (modified). $deg(v)$ denotes the degree of an intersection, i.e. the number of branches meeting at this intersection

GUARD works on a geographic network. This graph is annotated with information on landmarks, for example, their location and shape. The generation of context-specific route instructions is a three-step process. In the first step, for every decision point of the route, all instructions that unambiguously describe the route segment to be taken are generated, resulting in a set of possible instructions for each decision point. GUARD accounts for different types of landmarks in generating instructions whose role in the route instructions depends on their location relative to the route [15,19].

Next, GUARD performs *spatial chunking*. GUARD is flexible with respect to the principles used in these steps. For example, it allows integrating the chunking principles presented by Klippel et al. [8] or Dale et al. [3]. Finally, in the third step of GUARD, the actual context-specific route directions are generated. Here, from all possible instructions, those that best describe the route are selected. As this is realized as an *optimization* process, "best" depends on the chosen optimization criterion. Just as with the chunking principles, GUARD is flexible with respect to the criterion used. Optimization results in a sequence of chunks that cover the complete route from origin to destination. Due to the aggregation of instructions performed in chunking, instructions for some decision points will be represented implicitly, thus reducing the communicated information.

In summary, the approach to context-specific route directions finds the best instruction sequence according to the optimization criterion, but for a previously given route.

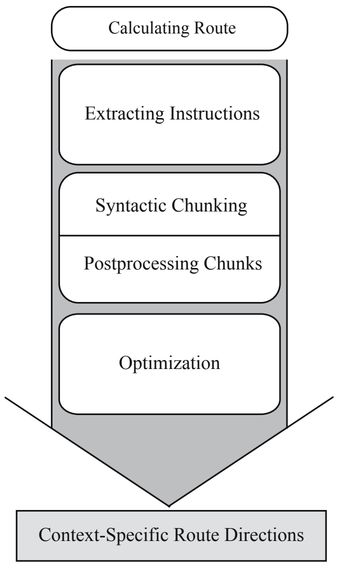


Fig. 2. Overview of GUARD, the generation process for context-specific route directions

3 Simplest Instructions Algorithm

In this section, the "simplest instructions" (SI) algorithm will be introduced. The algorithm combines the reasoning behind both simplest paths and context-specific route directions. Like simplest paths, the algorithm *finds* the best route, i.e., the route associated with the lowest cost in terms of instruction complexity. In that, the SI algorithm generates the best instructions for a route according to optimization criteria related to human direction giving, which are like those used in GUARD.

3.1 The Algorithm

The SI algorithm is based on Dijkstra's shortest path algorithm. Like Dijkstra's algorithm, the SI algorithm operates on a network represented as a graph $G = (V, E)$ comprising a set of vertices V and edges E connecting vertices, $E \subseteq V \times V$. Dijkstra's algorithm determines for each vertex in a graph the shortest path from a given start vertex (origin). It uses a cost function that determines the cost of traversing an edge. These costs are represented as the edges' labels. Starting from the origin, at each step the edge with the lowest costs is selected, which is then marked as visited. The costs for reaching all unvisited edges adjacent to the current vertex are then updated, i.e., it is checked whether the newly found path from origin to these edges is cheaper than the previously known one.

The SI algorithm, given in Algorithm 1 and explained in more detail below, differs from Dijkstra's algorithm in three key respects, considered in more detail in the following subsections:

1. the algorithm models the *instructions* required to describe a route, including the possibility of using landmarks in those instructions (section 3.2);
2. the cost function is associated with pairs of edges, rather than individual edges, to represent the cognitive cost of negotiating a decision point (section 3.2); and
3. the algorithm accounts for chunking of instructions, combining multiple instructions into one low cost (i.e., cognitively efficient) instruction (section 3.3).

3.2 Instructions and Costs

The *complete line graph* (or *evaluation mapping*) is the graph $G' = (E', \mathcal{E})$. E' is the set of edges in G, where the direction of edges is ignored (i.e., $(v_i, v_j) = (v_j, v_i)$ in E'). $\mathcal{E}$ is the set of pairs of vertices in E that share their "middle" vertex, i.e., $\mathcal{E} = \{((v_i, v_j), (v_j, v_k)) \in E \times E\}$ [20,16]. We refer to the elements of $\mathcal{E}$ as *decisions*. In other words, a pair of adjacent edges in E represents a decision an agent can take to move from one edge to the next.

The algorithm models the instructions required to describe a route as a set I of (arbitrary) labels. Instructions are associated with pairs of adjacent edges ("decisions"), describing a decision to move from one edge to another (e.g., "turn left at the intersection"). Instructions may include references to landmarks (e.g., "turn left at the post office"). Each pair of adjacent edges may have zero or more instructions associated with it (e.g., the instructions "turn left at the intersection" and "turn left at the post office" might both encode the same decision at a particular decision point). Each instruction may be associated with zero or more pairs of adjacent edges (e.g., "turn left at the intersection" might be a valid instruction for describing decisions at several different decision points). However, we assume no ambiguity in instructions, and disallow the possibility that the same instruction might be used to encode different decisions from the *same* edge (e.g., where "turn left at the intersection" can be used to describe more than one decision at a particular edge).

Formally, for the set of instructions I, the *labeling* function l is defined to be $l : \mathcal{E} \rightarrow I^2$. Thus, for a given pair of adjacent edges (e, e'), $l(e, e') = \{i_1, .., i_n\}$ gives the (possibly empty) set of instructions that describe that decision. Conversely, the *decision* function is defined to be $d : E \times \ I \ \rightarrow E \cup \ \{\varnothing\}$. For a given edge e and instruction i, $d(e, i) = e'$ gives the edge e' that results from executing the decision i at edge e. Note that e' may be the empty set (indicating that it is not possible to execute the decision i at edge e). Further note that since d is functional, we disallow ambiguity: there will be at most one edge $e' \in E$ that results from applying an instruction at edge e.

Each instruction has a cost associated with it using the function $w : I \rightarrow \ \mathbb{R}^+$ that models the cognitive effort associated with executing an instruction. Thus for a given instruction i, $w(i)$ yields the *cognitive cost* of executing instruction i. Potentially any cost function may be used, but in this empirical evaluation of the algorithm that follows we adopt the same cost function as previous work [17,19]. The SI algorithm minimizes the costs associated with traversing *pairs of* *edges* rather than individual edges (cf. [16]). In this way, the SI algorithm aims to minimize the *cognitive cost* of negotiating the decision points in a route, instead of minimizing the costs associated with travel (like distance or travel time).

3.3 Chunking

Spatial chunking is realized in the SI algorithm by spreading instructions forward through the graph from the edge currently being processed. Humans do not generate instructions with arbitrarily long chunks, so spreading instructions also needs to account for the cognitive and structural characteristics of the emerging chunks. This is implemented in a way that the approach is flexible with respect to the superordinate chunking principles used (Guard allows for similar flexibility in its chunking step). Considering, for instance, the principles discussed in [4] and [17], a chunk cannot be arbitrarily long unless a structural feature, such as a landmark, unambiguously marks its end.

For each pair of edges (e, e') we allow for a set of possible instructions. Allowing for multiple instructions for an edge corresponds to having a set of possible instructions to describe how to reach the next decision point from the current one. It is important to note that these multiple instructions do not correspond to having multiple edges between two vertices. Each instruction may have different costs associated with it and selection of an edge is determined by the instruction with the lowest costs. In the SI algorithm a vertex that has once been selected as the one with the lowest costs is never visited again in the search for the optimal path. If each instruction corresponded to an edge, then after selecting the one with the lowest costs all other edges connecting the two vertices would be unreachable. However, as discussed below, this would render transferring spatial chunking to the path-search algorithm impossible. That is, since a global selection criterion—spatial chunking—is introduced, the local selection criterion—choosing the next edge based on a instruction's costs—needs to account for more than one possibility to traverse this edge.

In terms of spreading instructions through the graph, a distinction needs to be drawn between an edge being *reachable* and being *chunkable*. An edge e_t is *reachable* from edge e_s with an instruction i if there exists a path from e_s to e_t that can be encoded as sequence of executions of instruction i. An edge e_t is *chunkable* from e_s with an instruction i if the sequence of i instructions required to reach e_t from e_s is also valid according to the employed superordinate chunking rules. As chunkable edges can be covered with a single instruction, the costs for all edges in a chunk are the same, namely those for reaching the first edge of a chunk using the chunked instruction.

An instruction needs to be spread forward as long as there are edges reachable with it. However, a cost update is only performed for those edges that are chunkable. This way of realizing spatial chunking can be viewed as dynamically introducing new edges to the graph that connect first and last vertex of a chunk. Figure 3 illustrates how instructions spread across neighboring edges and the distinction between reachable and chunkable edges. This cost updating is the reason why all instructions need to be considered when selecting the edge with the lowest

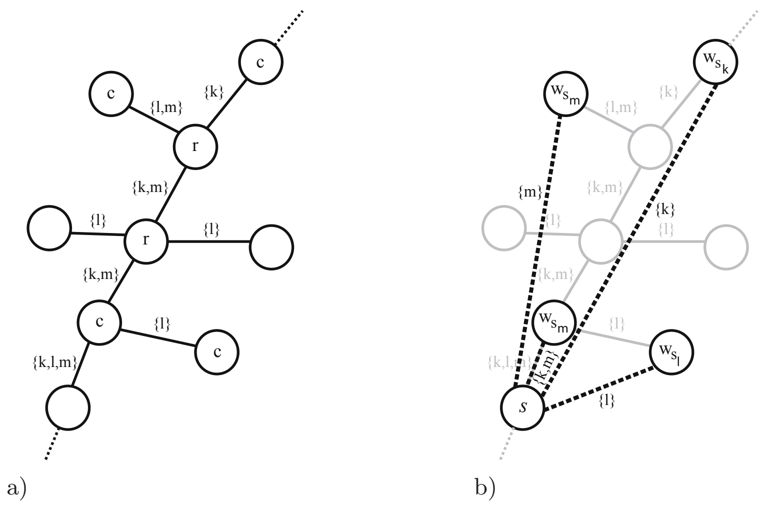


Fig. 3. Spreading instructions: a) If the instructions for reaching consecutive edges match, a vertex is *reachable*, denoted by r. If this combination also adheres to the superordinate chunking principles, a vertex is *chunkable*, denoted by c; the edges are labeled with their associated instruction sets (e.g., {k,m}); b) Chunkable vertices can be reached in a single step. This corresponds to dynamically introducing new edges between the first vertex of the chunk (denoted by s) and the chunkable one; reaching these vertices has the same costs as reaching the chunk's first edge (denoted by the different w_s).

costs. Globally, it might be less expensive to select an instruction that is locally more expensive if this instruction allows to cover more edges in the final path.

Formally, to implement this behavior in our algorithm we assume a *chunk validity* function $v : E \times E \times I \rightarrow \{true, false\}$. For a given start edge $e_s \in E$, terminator edge $e_t \in E$, and instruction $i \in I$, $v(e_s, e_t, i) = true$ only if e_t is chunkable from e_s using instruction i. We do not provide any further details of the actual procedure used to check chunk validity in this paper, since this issue is already covered in great detail in [4] and [17].

3.4 Algorithm Description

The SI algorithm is presented in Algorithm 1. In addition to the structures introduced above (the graph G, the complete line graph of G, the instruction set I, the labeling function l, the decision function d, the chunk validity function v) the algorithm requires an origin (starting) edge o as input.

The algorithm generates a predecessor function, $p : E \rightarrow E \times I$, that stores for each edge the preceding edge in the least cost path and the instruction used to reach the edge from its predecessor. If several edges are chunkable by an instruction, this predecessor is the first edge of the chunk (see Figure 3). Accordingly, when the algorithm visits an edge e, it needs to be checked for

Algorithm 1. Simplest instruction algorithm with multiple instructions and postponed chunking

Data: $G = (V, E)$ is a connected, simple, directed graph; $G' = (E', \mathcal{E})$ is the complete line graph of G; $o \in E$ is the origin (starting) edge; I is a set of instructions; $w : I \to \mathbb{R}^+$ is the instruction weighting function; $l : \mathcal{E} \to I^2$ is the labeling function; $d : E \times I \to E \cup \{\varnothing\}$ is the decision function; $v : E \times E \times I \to \{\mathit{true}, \mathit{false}\}$ is the chunk validity function.

Result: Function $p : E \to E \times I$ that stores for each edge the preceding edge and the instruction used in the least cost path.

```
1  // Initialize values;
2  forall $e \in E$ do
3      Initialize $c : E \to \mathbb{R}^+$ such that $c(e) \leftarrow \infty$;
4      Initialize $U_e \leftarrow \varnothing$;
5  Set $S \leftarrow \{\}$, a set of visited edges;
6  Set $p(o) \leftarrow (o, i)$ for some arbitrary $i \in I$;
7  Set $c(o) = 0$;
   // Process lowest cost edge until all edges are visited
8  while $|E \backslash S| > 0$ do
9      Find $e \in E \backslash S$ such that $c(e)$ is minimized;
10     Add $e$ to $S$;
11     forall $e' \in E \backslash S$ such that $(e, e') \in \mathcal{E}$ do
           // Update instruction/edge pairs from e to e'
12         forall $i \in l(e, e')$ do
13             $U_{e'} \leftarrow U_{e'} \cup \{(i, e)\}$;
14     forall $e' \in E \backslash S$ such that $(e, e') \in \mathcal{E}$ do
           // Update costs to e' based on instruction weights
15         forall $i \in I$ such that $d(i, e) = e'$ do
16             if $c(e') > c(e) + w(i)$ then
17                 Set $c(e') \leftarrow c(e) + w(i)$;
18                 Set $p(e') \leftarrow (e, i)$;
       // Perform chunking by propagating instructions forward
19     forall $(i, e_p) \in U_e$ do
20         Set $e_x \leftarrow e'$;
21         Set $X \leftarrow S \cup \{\varnothing\}$;
22         while $e_x \notin X$ do
23             $X \leftarrow X \cup \{e_x\}$;
24             Set $e_n \leftarrow d(e_x, i)$;
25             if $e_n \neq \varnothing$ then
26                 $U_{e_n} \leftarrow U_{e_n} \cup \{(i, e_p)\}$;
27                 if $c(e_n) > c(e')$ and $v(e_p, e_n, i) = \mathit{true}$ then
28                     Set $c(e_n) \leftarrow c(e')$;
29                     Set $p(e_n) \leftarrow (e_p, i)$;
30             $e_x \leftarrow e_n$;
```

every instruction holding for (e, e') whether it lowers the costs associated with e', and whether it can be used to reach other edges from e' as well (Algorithm 1, lines 1–1). Chunking (i.e., the updating of weights to edges other than those directly connected to the current edge) can be postponed until after the edge weights have been updated (Algorithm 1, lines 1–1). In order to keep track of which instructions need to be further considered at an edge, all instructions the edge has been reached by so far need to be stored, along with the edge the current edge has been reached on (for correctly setting predecessors). Formally, for each edge, the algorithm also stores a set U_e of instruction/edge pairs (i, e), initialized in line 1.

Next, the algorithm updates the weights associated with the unvisited edges that are incident with the current edge e (Algorithm 1, lines 1–1). This step is as used in the simplest path algorithm [16], and is essentially the same core condition used in Dijkstra's algorithm operated upon the complete line graph.

Finally, Algorithm 1, lines 1–1 performs chunking by propagating instructions forward from the current edge as far as is possible. As already discussed, all edges that are reachable store information about the instruction being forward propagated (line 1), but only those edges for which the resulting instruction would be a valid chunk have their weights updated (lines 1–1).

3.5 Computational Time Complexity Analysis

The computational time complexity of finding the lowest cost path through the graph is $O(|E|^2)$, because in the worst case for each edge visited the algorithm must update the costs of every other edge. Similarly, the computational cost of spreading the selected instruction through the graph is $O(|E|^2)$, because it requires visiting each edge in turn and in the worst case, spreading instructions to every other edge in the graph. Thus the overall time complexity of the algorithm is $O(|E|^2 + |E|^2) = O(|E|^2)$. Compared to Dijkstra's shortest path algorithm, which is $O(|V|^2)$, in a totally connected graph with $|E| = |V|^2$ edges, this results in an overall time complexity of $O(|V|^4)$.

However, as argued in [16], geographic routing problems never deal with totally connected graphs. If instead a planar graph is assumed, then by Euler's formula (simple connected planar graph has n vertices - m edges + f faces = 2) the number of edges is linear in the number of nodes, $|E| \leq 3(|V| - 2)$. Thus, the overall time complexity for the SI algorithm applied to a planar graph is $O(|V|^2)$. This is the same as the complexity of Dijkstra's algorithm, so we conclude the SI algorithm does not increase the computational time complexity compared with the simplest path or shortest path algorithms, at least for planar graphs.

3.6 Reconstructing Routes

Algorithm 1 generates a predecessor function $p : E \rightarrow E \times I$ that stores for each edge the preceding edge in the least cost path and the instruction used to reach the edge from its predecessor. Reconstructing a path to a particular destination edge t is then simply a matter of backtracking from t using p, at each step storing

Algorithm 2. Algorithm for reconstructing the simplest instruction path

```
Data: $G = (V, E)$ is a connected, simple, directed graph; $o \in E$ is the origin
      (starting) edge; $t \in E$ is the target (destination) edge;
      $d : E \times I \to E \cup \{\varnothing\}$ is the decision function; $p : E \to E \times I$ is the
      predecessor function (generated by algorithm 1).
Result: A sequence (word) $P$ of edges corresponding to the optimal path and a
        sequence (word) $L$ of labels corresponding to the best sequence of
        instructions.
1  Set $P$ to be the empty word $\lambda$;
2  Set $L$ to be the empty word $\lambda$;
3  Set $T$ to be the empty word $\lambda$;
4  Set $e \leftarrow t$;
5  while $e \neq o$ do
6      Let $p(e) = (e_p, i)$;
7      Set $L \leftarrow i + L$;
8      Set $e' \leftarrow e_p$;
9      Set $T \leftarrow e'$;
10     while $(e', e) \notin E$ do
11         $e'' \leftarrow d(e', i)$;
12         Set $T \leftarrow T + e''$;
13         Set $e' \leftarrow e''$;
14     Set $P \leftarrow T + P$;
15     Set $e \leftarrow e_p$;
```

the predecessor edge and instruction in a list. Algorithm 2 gives an example of reconstructing routes using p (using an algebraic language notation, where E and I form alphabets, and the lists of edges and instructions in the route are stored as sequences P and L of letters—words—from those alphabets, constructed by iteratively prepending letters to the initially empty word with the concatenate + operator). In Algorithm 2, retrieval of the instruction word L simply requires direct backtracking through the predecessor list (line 2). Construction of the edge list word P requires an additional loop to retrieve the edges between chunked instructions (lines 2–2).

4 Comparisons

As we have demonstrated in the last section, the SI algorithm is able to incorporate fundamental principles of human direction giving without increasing the overall computational complexity of the route generation algorithm. In addition to computational complexity, other measures of the performance of the SI algorithm are the length of paths the algorithm generates, and the number of instructions required to describe the route.

The length of the path described by the simplest instructions is necessarily equal to or longer than the shortest path. Thus, the length of the simplest instruction path, when compared with the shortest path, provides a measure of

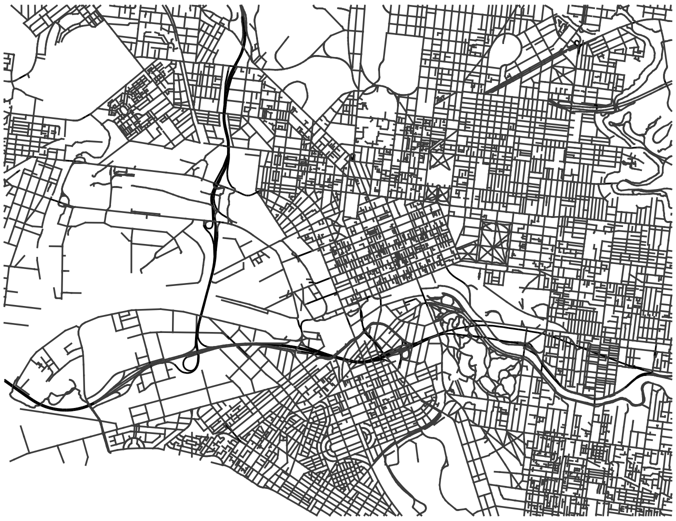


Fig. 4. The test area: part of Melbourne, Australia

the detour a wayfinder would need to take when using the simplest instruction path. Conversely, the length of instructions generated by the SI algorithm is expected to be shorter that those generated by the shortest path (assuming one instruction per decision point).

To analyze these aspects, the lengths of paths and instruction sequences produced by the SI algorithm are compared with those produced by Dijkstra's shortest path as well as the simplest path algorithm. Thus, these results aim to provide an indication of the balance struck by the different algorithms between the desire for direct (short) routes and simple (short) route instructions.

4.1 Data

The algorithm's performance was tested using several different geographic data sets. The results in this section were derived from a transportation network data set representing part of the inner city (CBD) and surrounding districts of Melbourne, Australia (see Figure 4). The transportation network was augmented with objects representing landmarks. The landmark objects in this case were derived from the railway infrastructure in the area (essentially they are train stations). Clearly, railway infrastructre will not always be appropriate for human wayfinding, but provides an adequate simplification in the context of the following experiments.

4.2 Results of Experiments on Path Length

All three algorithms—shortest path, simplest path, simplest instructions—were used to calculate paths between randomly chosen origins and destinations in

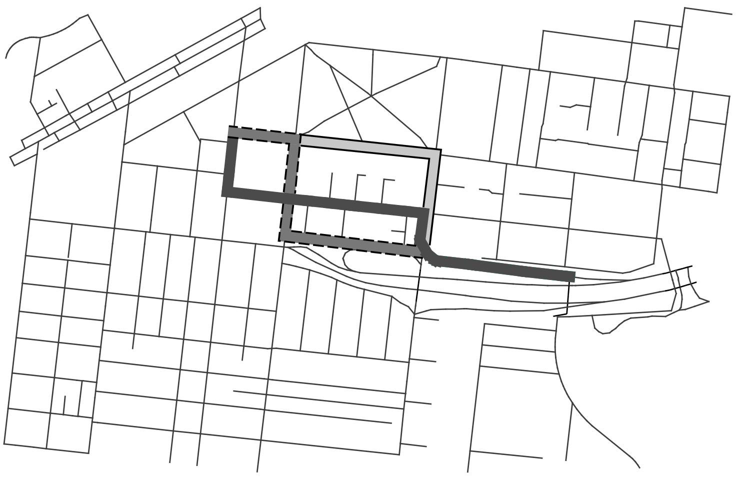


Fig. 5. Shortest path (the thick black line), simplest path (the gray line), and SI path (the dark gray line with dashed border) for a sample origin / destination pair

the network. For each origin/destination pair, each algorithm calculated a path. For each of 53,000 different origin/destination pairs tested, the resulting paths from each algorithm were compared in terms of path length and the number of associated instructions required to describe the path (in simplest and shortest paths, the number of decision points; in simplest instructions the number of chunked instructions). Figure 5 depicts typical differences in shortest, simplest, and simplest instruction paths between two points.

On average, paths determined with the SI algorithm were 13.31% longer than the shortest path between origin and destination. Using simplest paths, there was an increase in length of 12.52% (a result comparable to [16], which found lengths of simplest paths that on average were 15.8% longer than the corresponding shortest path). The average path length of simplest paths and simplest instructions were almost equal (3,645.82m to 3,671.49m, standard deviation of 1,821.49m and 1,840.99m, respectively).

A *t*-test was conducted to test the hypothesis that the simplest instruction paths are longer than the simplest paths. The test showed that the differences in length were significant at the 1% level, so we conclude that the SI algorithm does generate paths that are longer than the simplest path. However, while the differences were significant, a test for effect size results in very small differences, 0.06. This value indicates that the actual differences in length between simplest paths and simplest instruction paths are very small; the average increase in length for paths generated by the SI algorithm is 25.67m.

With respect to the shortest paths, the simplest instruction paths were again significantly longer than the corresponding shortest path, as expected. In more detail, from all 53,000 paths, 40,232 SI paths were less than 15% longer than the corresponding shortest (with 6829 paths being equally long to the shortest path); 3481 were more than 25% longer than the shortest path; only for 54 paths was this increase more than 50%. In summary, for only 6.7% of all cases were the simplest instruction paths more than 25% longer than the shortest path; in 75.9% of all cases the simplest instructions paths were less than 15% longer than the shortest path.

4.3 Results of Experiments on Instruction Length

The algorithm for simplest instructions significantly reduces the number of instructions needed to descibe the routes generated. Assuming an instruction is required for every decision point in shortest and simplest paths, the number of instructions required to describe the simplest instruction paths was on average average 57.93% less than required for shortest paths, and 55.37% less than required for simplest paths. A Wilcoxon signed-rank test confirms that this difference is statistically significant at the 1% level. This result is comparable to previous evaluations done with Guard and illustrates the strengths of spatial chunking even when employed on data sparsely annotated with landmarks. The result also highlights that while simplest paths can help to minimize the cognitive complexity of individual instructions used in the route, they do not necessarily provide savings in terms of the global number of instructions required to describe the route. By contrast, the paths generated by the SI algorithm do dramatically reduce the overall length of route instructions.

One final test was to compare the length of the instructions generated by the SI algorithm with the length of instructions generated using Guard applied to the corresponding shortest path. As expected, in all cases the length of the instructions generated by the SI algorithm was equal to or less than the corresponding chunked instructions generated by Guard applied to the shortest path. We used the same Wilcoxon-test as before to test the difference for stastitical significance. It shows that there is a significant advantage in using the SI algorithm over simply applying Guard to the shortest paths.

5 Conclusions

We have presented the simplest instructions (SI) algorithm, which is based on Dijkstra's shortest path algorithm. The SI algorithm integrates fundamental principles of human direction giving, namely references to landmarks and spatial chunking, in finding a route between an origin and a destination. Multiple labels attached to an edge capture several options to describe which action to perform for reaching the next decision point. Generation of these labels is based on Guard, a process for producing cognitively ergonomic route instructions. Spatial chunking, i.e., the subsumption of several consecutive instructions to a single one, is realized by spreading labels forward through the graph and dynamically introducing new edges. The SI algorithm computes paths in the same order of time complexity as the generalized Dijkstra algorithm, $O(n^2)$.

An empirical evaluation, comparing the path lengths produced by the SI algorithm with the corresponding shortest and simplest paths, has shown promising results. On average, the SI paths are about 13% longer than the shortest path, which is in the same range as the increase in length introduced by simplest paths. The length of instructions, however, is decreased by 57%, i.e., on average slightly more than two consecutive instructions can be chunked into a single one, reducing the amount of information that needs to be communicated by more than half. Thus, with only a slight increase in path length, the SI algorithm produces instructions that can be expected to be significantly easier to follow.

For a small number of paths, however, there is a considerable increase in path length. To counter these cases, the algorithm may be adapted to balance the increase in path length and the ease of instructions (see also [21]). Here, human subject studies are called for to elicit sensible parameters. Furthermore, as spatial chunking relies to a good part on the presence of landmarks, a more detailed analysis of the relationship between the density of landmarks on one hand, and path length and chunking ratio on the other hand will reveal a more detailed picture of the algorithm's performance and may point out refined methods in choosing labels and applying chunking. Also, an analysis of how an environment's structure influences the resulting paths may allow identifying strategies on how instructions may be automatically adapted to different environmental situations.

Acknowledgments

Kai-Florian Richter's research is supported by the Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition, which is funded by the Deutsche Forschungsgemeinschaft (DFG). Matt Duckham's research is partially supported under the Australian Research Council's Discovery Project funding scheme (project number DP0662906). Matt Duckham is grateful for the support of Ordnance Survey. We like to thank Holger Schultheis for discussing parts of the paper and appreciate comments of three anonymous reviewers.

References

1. Tversky, B., Lee, P.U.: Pictorial and verbal tools for conveying routes. In: Freksa, C., Mark, D.M. (eds.) COSIT 1999. LNCS, vol. 1661, pp. 51–64. Springer, Heidelberg (1999)
2. Habel, C.: Incremental generation of multimodal route instructions. In: Natural Language Generation in Spoken and Written Dialogue, AAAI Spring Symposium 2003, Palo Alto, CA (2003)
3. Dale, R., Geldof, S., Prost, J.P.: Using natural language generation in automatic route description. Journal of Research and Practice in Information Technology 37(1), 89–105 (2005)
4. Klippel, A., Tappe, H., Kulik, L., Lee, P.U.: Wayfinding choremes — a language for modeling conceptual route knowledge. Journal of Visual Languages and Computing 16(4), 311–329 (2005)
5. Dijkstra, E.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)

6. Denis, M.: The description of routes: A cognitive approach to the production of spatial discourse. Cahiers Psychologie Cognitive 16(4), 409–458 (1997)
7. Lovelace, K.L., Hegarty, M., Montello, D.R.: Elements of good route directions in familiar and unfamiliar environments. In: Freksa, C., Mark, D.M. (eds.) COSIT 1999. LNCS, vol. 1661, pp. 65–82. Springer, Heidelberg (1999)
8. Klippel, A., Tappe, H., Habel, C.: Pictorial representations of routes: Chunking route segments during comprehension. In: Freksa, C., Brauer, W., Habel, C., Wender, K.F. (eds.) Spatial Cognition III. LNCS (LNAI), vol. 2685, pp. 11–33. Springer, Heidelberg (2003)
9. Hirtle, S.C., Jonides, J.: Evidence of hierarchies in cognitive maps. Memory & Cognition 13(3), 208–217 (1985)
10. Couclelis, H., Golledge, R.G., Gale, N., Tobler, W.: Exploring the anchor-point hypothesis of spatial cognition. Journal of Environmental Psychology 7, 99–122 (1987)
11. Michon, P.-E., Denis, M.: When and why are visual landmarks used in giving directions? In: Montello, D.R. (ed.) COSIT 2001. LNCS, vol. 2205, pp. 400–414. Springer, Heidelberg (2001)
12. Raubal, M., Winter, S.: Enriching wayfinding instructions with local landmarks. In: Egenhofer, M., Mark, D. (eds.) GIScience 2002. LNCS, vol. 2478, pp. 243–259. Springer, Heidelberg (2002)
13. Elias, B.: Extracting landmarks with data mining methods. In: Kuhn, W., Worboys, M., Timpf, S. (eds.) COSIT 2003. LNCS, vol. 2825, pp. 375–389. Springer, Heidelberg (2003)
14. Caduff, D., Timpf, S.: The landmark spider: Representing landmark knowledge for wayfinding tasks. In: Barkowsky, T., Freksa, C., Hegarty, M., Lowe, R. (eds.) Reasoning with mental and external diagrams: computational modeling and spatial assistance - Papers from the 2005 AAAI Spring Symposium, Menlo Park, CA, pp. 30–35 (2005)
15. Hansen, S., Richter, K.F., Klippel, A.: Landmarks in OpenLS - a data structure for cognitive ergonomic route directions. In: Raubal, M., Miller, H., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 128–144. Springer, Heidelberg (2006)
16. Duckham, M., Kulik, L.: ”Simplest” paths: Automated route selection for navigation. In: Kuhn, W., Worboys, M., Timpf, S. (eds.) COSIT 2003. LNCS, vol. 2825, pp. 169–185. Springer, Heidelberg (2003)
17. Richter, K.-F., Klippel, A.: A model for context-specific route directions. In: Freksa, C., Knauff, M., Krieg-Brückner, B., Nebel, B., Barkowsky, T. (eds.) Spatial Cognition IV. LNCS (LNAI), vol. 3343, pp. 58–78. Springer, Heidelberg (2005)
18. Mark, D.: Automated route selection for navigation. IEEE Aerospace and Electronic Systems Magazine 1, 2–5 (1986)
19. Richter, K.F.: A uniform handling of different landmark types in route directions. In: Winter, S., Duckham, M., Kulik, L., Kuipers, B. (eds.) COSIT 2007. LNCS, vol. 4736, pp. 373–389. Springer, Heidelberg (2007)
20. Winter, S.: Weighting the path continuation in route planning. In: GIS 2001: Proceedings of the 9th ACM international symposium on Advances in geographic information systems, pp. 173–176. ACM, New York (2001)
21. Haque, S., Kulik, L., Klippel, A.: Algorithms for reliable navigation and wayfinding. In: Barkowsky, T., Knauff, M., Ligozat, G., Montello, D.R. (eds.) Spatial Cognition 2007. LNCS (LNAI), vol. 4387, pp. 308–326. Springer, Heidelberg (2007)

Road Networks and Their Incomplete Representation by Network Data Models

Simon Scheider[1] and Werner Kuhn[2]

[1] Fraunhofer Institut für Intelligente Analyse und Informationssysteme (IAIS), Schloss Birlinghoven, 53754 Sankt Augustin, Germany
simon.scheider@iais.fraunhofer.com
[2] Institute for Geoinformatics (ifgi), University of Münster, Weselerstr. 253, 48151 Münster, Germany
kuhn@uni-muenster.de

Abstract. Road networks, roads, and junctions are examples of natural language terms whose semantics can be described by affordances of their physical referents. In order to define affordances in such a way that they can be used for classifying and describing instances in a geographic database, one has to deal with the problems of informational incompleteness and limited definability. In this paper, we propose an affordance-based theory of channel networks, based on the work of Hayes [4], as a means to derive necessary conditions for database representations of road networks. By exploring this example, we show that affordance-based logical definitions are a convenient method to capture essential properties of physical objects usually not present in their database representation, but appropriate to explain and define its structure.

Keywords: Ontology, Road network, Affordance-based theory, Naïve physics.

1 Physical Object Notions and Geographic Databases

Affordance-based logical theories were inspired by Gibson's work [2] and have been used as a means for semantic analyses [7][8][12]. But can they also be used to classify representations of physical objects in a *geographic database* into affordance-based categories? For example, can an affordance-based specification of a certain kind of intersection be used to classify features in a road database? If the answer is positive, this provides an approach to semantically annotate the contents of databases and services.

We assume that a formal definition of a natural language term in a logical theory specifies *a human category* denoted by this term. Since each feature is an instance of a certain data type (a.k.a. feature type), we can focus on the evaluation of a data model with respect to category definitions. There are two major challenges in this approach:

Limited Definability. Attempts to formally define terms of a natural or technical language face the "human knowledge soup" [11], so they suffer from overgeneralizations, abnormal conditions, unanticipated applications, and incomplete definitions. This usually means that (1) existing instances of a category are overlooked ("birds fly", but what about penguins?) and (2) non-instances are included ("birds fly", is then a bat also a bird?). Although this problem is generally unsolvable, it does not mean that incomplete definitions are useless. Ontologies are in fact meant to be partial specifications of conceptualizations [3]. Furthermore, as Hayes [5] has pointed out, a logical theory aiming to describe the physical world can be at most *experimentally complete* (see Israel [6]) or *conceptually closed*, so that everything that should be said can be said, and it is always doomed to include "ghostly" *unintended models* in its universe of meaning. One can handle this problem by tying the meaning of the theory's tokens to observational systems [5]. In our case we try to tie a physical theory to a data model, but this poses another severe problem.

Informational Incompleteness. Essential properties of the theory are not explicitly represented or excluded in the data model. Think about a computer application that has to figure out where to "go and get a coffee" by using a geometric data model of all objects in a room. One must deal with the difficulty of *how to interpret a theory in an incomplete data model*, because essential aspects of the task, like for example the information about the location of liquids in a room, are not present in it. This difficulty could be solved if the intelligent agent knew that liquids can be contained in a cup and if it could identify a cup. Therefore a solution to this problem can be appropriate knowledge representation [4]. For this we suggest the usage of *an affordance-based theory*, which allows to partially infer the missing types and their properties.

In this paper, we develop a conceptually closed theory of *movements and supports* in which it is possible to define the category *road network by describing what actions its members must afford*. From this definition it is then possible to *infer and explain necessary structural properties of any road network data model*. Our work presents a method for defining arbitrary categories of a road network database by explaining its structural properties using a specialized affordance theory. We intend to address more sophisticated examples like "road" and "junction" explicitly in the near future. As current data model standards for transportation networks, like ISO GDF, are rather informal descriptions, we furthermore see this work as a contribution to a formal domain ontology.

In the next section, we sketch our approach. In section 3, we introduce a common graph theoretic data model for navigational road network data. In section 4.1, we introduce the abstract concept of a history, which is the basic thing to exist in a physical world that has to cope with movements. A purposefully developed ontology of channels is introduced in section 4.2. In section 5, we introduce an affordance-based notion of a channel network and partially interpret the network theory in the graph data model.

2 Methodological Sketch

Affordances say something about the kinds of *physical actions* that are possible in an environment. But affordances also say something about certain *static aspects of the environment*, which we call *structure*. Physical actions and structures are thought to be mutually dependent.

For our purpose, we need an affordance-based logical theory consisting of basic and derived types, each of which denotes a set of objects in the theory's universe, and

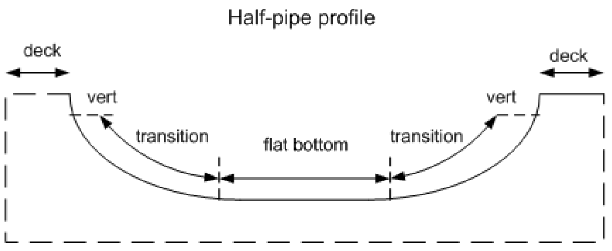


Fig. 1. A half-pipe in a skateboard world is an example of an affordance category

some axioms and definitions that give meaning to them. We suppose there are two non-functional kinds of types *structure* and *action.* For our affordance-based theory to be useful, it should *logically restrict* the joint appearance of structures and actions in the *conceptual affordance relation*, such that if we restrict the structure part of the theory, the action part is restricted as a consequence, and vice versa. For example, if our physical universe of traffic had only rollerblades and skateboards as conveyances, then the whole accessible world's infrastructure would be restricted to 2 1/2 dimensional solid objects with smooth surfaces.

Our task now is to define *a category as a structure subtype* in terms of the *commonsense knowledge* that comes with it, and then to interpret this definition in a data model. For example, in our skateboard world, we would like to define a "half-pipe" (compare FIGURE 1) in order to search for "half-pipe candidate" objects. We suppose that affordance-based theories are especially well suited for this purpose, because on the one hand, *the type of afforded action is usually what we know a-priori* about such a category, so it comes close to our commonsense knowledge about it. As a skateboarder, we perfectly know that a half-pipe is a thing that allows us to attain extreme speeds by an action called "pumping". On the other hand, we can assume that every member of such a category must afford this type of action, and therefore we have a convenient means of finding *necessary conditions* for an object to be in that category. So we can safely call every object that affords pumping actions a half-pipe. The most important aspect is nevertheless that the affordance-based theory allows us to infer what is usually not known a-priori for the category, which are the *structural properties* of the object that exist due to its afforded action. If we search for a half-pipe in our skateboard world, we may infer that everything which is similar to a certain upward concavity could be one. It is therefore possible to close the informational gap between incomplete knowledge about afforded actions and an incomplete data model of the affordance structure.

An affordance-based theory allows defining *a physical object in affordance terms,* that is defining its structural properties with the help of its afforded actions. These properties are expressed by the set of theorems about the defined structural subtype, which we call *affordance theorems.* A data model is said to *partially satisfy* the affordance definition, if there is a translation from every non-logical symbol of this model into the theory such that the translated tokens all satisfy the subset of affordance theorems that exist about them. We call this test *weak data satisfiability of a data model.*

The proposed method is considered to be useful in at least two respects (much in the spirit of analytic/deductive machine learning):

- In order to find reliable and operational (but necessarily incomplete) formal definitions of categories for databases
- In order to find domain theoretical explanations for structural properties of data base instances

In the remainder of the paper, we show that the data model introduced in the next section, which is a common representation of a road network (for details see [10]), satisfies the structural part of the affordance based definition of a road network developed in section 4 and 5.

3 A Common Road Network Data Model

Let the *road network* $S = (N, L)$ be a *directed graph* with *nodes* N and *edges* $L \subseteq N \times N$. For an edge $l = (n_1, n_2)$, let $l^- = n_1$ its incident *source* and $l^+ = n_2$ its *sink*. A node stands for a street intersection and an edge for one direction of a street segment between two intersections. Note that a bidirectional street segment is represented by two *parallel edges* $l = (n_1, n_2)$ and $l' = (n_2, n_1)$. A *trace* t consists of a sequence of edges $l_1, \ldots, l_n$ such that $l_i^+ = l_{i+1}^-$ for $all\ i = 1, \ldots, n-1$. Let $t^- = l_1^-$ denote the *source* and $t^+ = l_n^+$ the *sink* of the trace t.

We call a road network *embedded* if we additionally have mappings $point : N \rightarrow P$ and $line : L \rightarrow 2^P$ where P is the set of points in the Euclidian plane; the embedding is a simple linear curve in the Euclidian plane that geometrically connects the points of its two incident nodes. We assume that bidirectional street segments have the same geometry, i.e. $line(n_1, n_2) = line(n_2, n_1)$.

Further, we refer to a subset $E \subseteq \{(l_1, l_2) \in L \times L \mid l_1^+ = l_2^-\}$ as *navigation relation* for vehicle traffic (compare *line graphs L(S)* [1] or *linear dual graphs* [14]): $(l_1, l_2) \in E$ implies that it is possible (according to traffic rule and street construction) to drive from the segment l_1 into the segment l_2 crossing their incident node $n = l_1^+ = l_2^-$. A trace $t = l_1, \ldots, l_n$ of the road network $S = (N, L)$ is *navigable* iff $(l_i, l_{i+1}) \in E$ for all $i = 1, \ldots, n-1$. Note that navigable paths or traces are *a true subset* of all the paths in S, because there are certain junction types where not each possible path in S is allowed (compare Scheider et al. [10] and FIGURE 2).

We henceforth assume that S is *connected by navigable traces*, i.e. for all nodes n, $n' \in N$, there exists a navigable trace t such that $t^- = n$ and $t^+ = n'$.

From this several properties follow. We call a network edge in L *graveyard* if it has no outgoing navigation edge in the navigation relation E, that is $\{l \mid \neg\exists l_2.(l, l_2) \in E\}$. We call a segment a *factory* if it is an element of the set $\{l \mid \neg\exists l_1.(l_1, l) \in E\}$,. Because S is *connected by navigable traces*, we can exclude graveyards and factories in S, that is, each network edge in S must both have at least one navigation tuple leaving the segment and one navigation tuple entering the segment. Furthermore, each node must have two appropriately directed adjacent segments in S being a navigable tuple from E, and hence the *minimum degree of each vertex* in S is greater than or equal to 2. So $\forall n \in N.\ \exists (l_1, l_2) \in E.\ l_1^+ = l_2^- = n$.

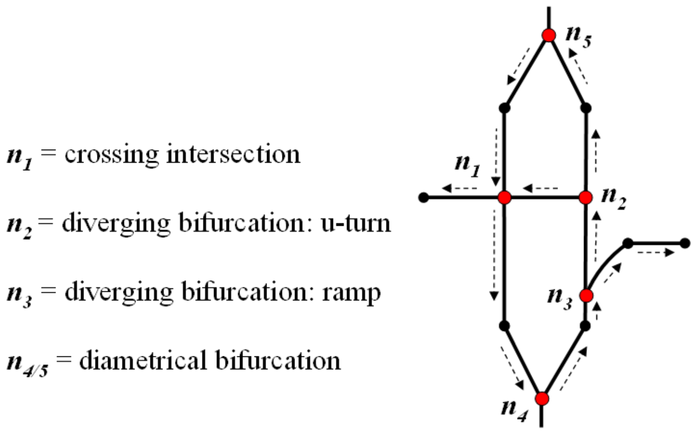

n_1 = crossing intersection
n_2 = diverging bifurcation: u-turn
n_3 = diverging bifurcation: ramp
$n_{4/5}$ = diametrical bifurcation
n_5
n_1
n_2
n_3
n_4

Fig. 2. Types of one-way intersection nodes for a dual carriageway road. Diametrical bifurcations ($n_{4/5}$) are examples of nodes with navigational restrictions.

For illustration purposes, let us consider some common road network features in this model. All non-parallel network edges in *S* are called *one-way* edges. Navigation tuples connecting two bidirectional network edges are called *u-turns*. A set of two bidirectional network edges with one of its two vertices exclusively being part of a *u-turn* edge, is called *dead-end*. Vertices in *S* can be e.g. *diverging* or *converging bifurcations*, *diametrical bifurcations* and *crossings* (compare FIGURE 2). Definitions of the higher level concepts *bidirectional road*, *dual carriageway* and *junction* (e.g. roundabout) are straightforward in this data model, compare [10].

4 What Road Networks Afford

The notions *structure* and *action* are domain dependent key concepts of every affordance definition, and therefore need a formal treatment consistent with our domain knowledge about the category. In the following theory, we elaborate such formalization for road networks in terms of the types *flat support* and *supported movable history*.

4.1 Histories and Movements

We base our theory on the logic of histories (for a detailed discussion we refer to [4]), and extend Hayes' theory to derive an ontology of movements, channels and networks.

Some remarks on the notational style. Sentences are written in predicate logic. For convenience, every expression is defined because invalid function applications have an *undefined* value and invalid predicate applications are defined to become false. Variables and constants are implicitly typed. Free variables are understood to be universally quantified. $\exists!$ means the existence of exactly one instance. We always presuppose $a \neq b$ for two different symbols. Predicates are capitalized. Additionally, we use the ordinary operations on (point) sets and the usual Euclidian vector calculus. Some definitions and proofs are omitted and informally described in the text.

The principal argument of Hayes is, that in order to identify physical objects (and the things they are made of), we can cut space-time into *histories*. We describe histories as pieces of 4-dimensional space-time, $h \subset R^4$ (see appendix). We assume that histories and their lower dimensional projections in geographic space G and time T have well defined *dimensionality*, *boundaries* and *interiors*. A formal definition is given in the appendix. We can think of *point histories* as 0-dimensional, *curve histories* as 1-dimensional, *surface histories* as 2-dimensional and *regular histories* as 3-dimensional compact entities in geographical space that are "one-piece" and exist for an extended time interval. Note that the definition of the boundary and interior of a history h, denoted by ∂h and $h°$, contrasts sharply with usual definition in point set topology.

We furthermore adopt the notion of a *face*[1], and the function *toso* meaning that a piece-of-space is to the other side of a face of another piece-of-space (see appendix). We will use the binary predicate *Disjoint* to express that two pieces of space do not intersect, and *Joined* if they have just a face in common. We say that two pieces of space are *InContact* if they are *Disjoint* but there is only a free surface between the two (compare [4]).

A *state* is an *instantaneous spatial slice* of a history at a certain time-instant, that is the projection of a history h into G for a time instant t. A certain state is denoted by $h@t$. As histories are *topologically connected and bounded*, there are two uniquely defined states called *start* and *finish*. Consequently, there are two unique time instants called *begin* and *end* (compare FIGURE 4).

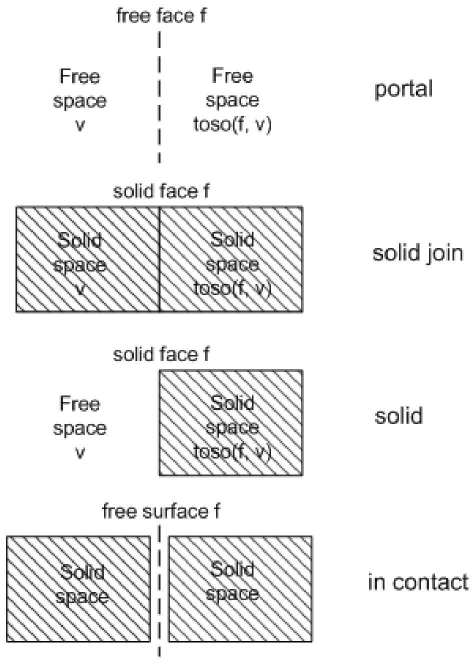


Fig. 3. Free and solid pieces-of-space and their faces, adapted from [4]

[1] A face of a piece-of-space A is a piece-of-space F of one lower dimension which forms part of A's boundary.

Histories are not arbitrary subsets of Euclidian space; they are always thought to *contain a physical object in each time instant*. This object is different from the *piece-of-stuff* or the *composite* it is *made of*: we identify a river not by identifying the water in it, and we identify a car not by identifying its parts. We call a history *solid*, if its object is made of the same things in each moment in time, so that other *histories cannot intersect with its interior*. In order to specify this, we introduce the predicate *Free* and appropriately constrain the topological relations of solid (non-free) histories:

Axiom 1. $History(h_1) \wedge History(h_2) \wedge \neg Free(h_1) \wedge \neg Free(h_2) \Rightarrow$
$(\forall t_1 \in when(h_1), t_2 \in when(h_2). Joined(h_1@t_1, h_2@t_2)) \vee$
$(\forall t_1 \in when(h_1), t_2 \in when(h_2). Disjoint(h_1@t_1, h_2@t_2))$

We furthermore assume that a free history does not include its faces, so it is equal to its interior (and therefore its faces can be either solid or free), whereas a solid history does (compare FIGURE 3).

When(h) denotes the time interval during which h takes place. *Where(h)* denotes the spatial projection of *h* in *G* for all time instances, that is the place where the history "takes place".

An *episode e* of a history *h*, *episode(e, h)*, is defined as a subset of *h* which has the same spatial extensionality as *h* for a non-zero subinterval of *when(h)*:

Definition 1. $Episode(e,h) \Leftrightarrow History(e) \wedge when(e) \subseteq when(h) \wedge \forall t \in when(e). h@t=e@t.$

A history is said to be *rectangular*, if it doesn't change its geometric state in time:

Definition 2. $Rectangular(h) \Leftrightarrow \forall t_1, t_2 \in when(h). h@t_1 = h@t_2.$

A *history is isometric* if each pair of spatial points has a fixed Euclidian distance over all states. Let f_I be a congruence isomorphism[2] (isometric) function on Euclidian space R^3 with distance function[3] *d*. A history is said to be *isometric* if it has a congruent shape for each pair of time instants:

Definition 3. $Isometric(h) \Leftrightarrow \forall t_1, t_2 \in when(h). \exists f_I. \forall s_1 \in h@t_1. \exists s_2 \in h@t_2. f_I(s_1) = s_2.$

Clearly, rectangular(h) $\Rightarrow$ isometric(h). The function $Traj_{x/t}^{h} = \{<t_i, f_I^{t, ti}(x)> | \; t_i \in when(h)\}$ for some fixed $x \in h@t$ and some fixed $t \in when(h)$ denotes the *trajectory of point x at time t* of the isometric history *h*. Let point *x* be the object's centroid. Clearly, as trajectories are point histories, they have episodes, start and finish, begin and end. Furthermore, we call the spatial projection of a trajectory, where(Traj), its *path* (compare FIGURE 4).

Now we can define the notion of *differentiability* for isometric histories. We call the first vector derivative of a trajectory the *velocity vector* of that time instant, $v_f(t)$. Differentiability is defined for all time points in the open time interval (denoted by]…[) that the history encloses:

[2] That is, a combination of a linear translation and a rotation preserving relative distances between points.

[3] Function f_I is isometric iff $\forall$ x, y $\in$ G. $d(f_I(x), f_I(y)) = d(x,y)$.

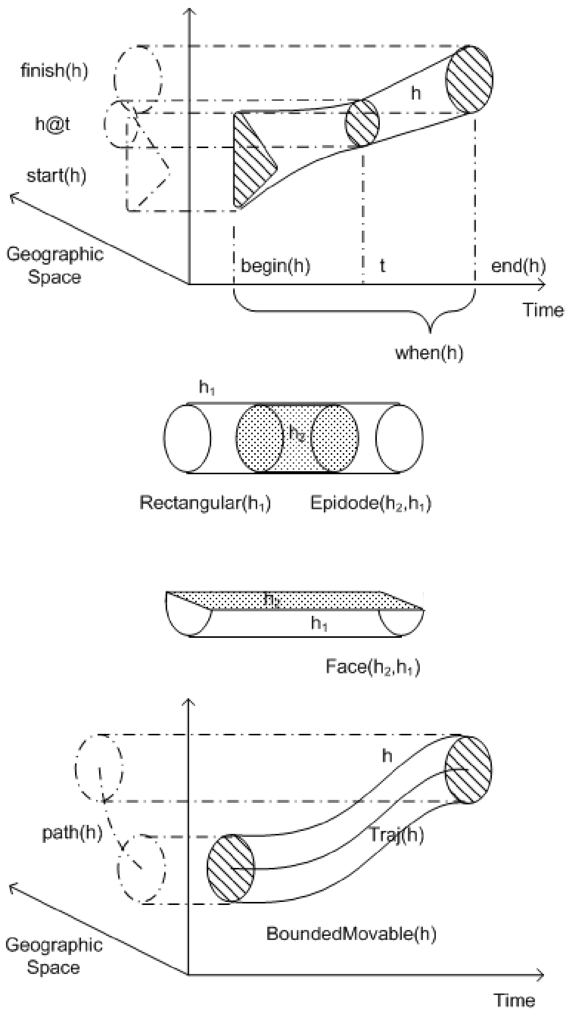


Fig. 4. Illustration of histories. Adapted from [4].

Definition 4. *Differentiable(h)* $\Leftrightarrow$ *Isometric(h)* $\wedge$ $\forall t \in]begin(h);end(h)[.\ \lim_{\Delta t \to 0}(Traj^h(t+\Delta t) - (Traj^h(t)+v_{Traj}(t)*\Delta t))/\Delta t = \mathbf{0}$.

This has the consequence that the trajectory does not have any "sharp corners" with sudden jumps in direction. Also, from *differentiability* follows *continuity* for the open time interval of history *h*.

Movable histories obviously enclose solid objects, which is the reason for their differentiable behavior in time. In naïve physics, the histories of solid objects cannot finish at once [4], because they first have to lose their form. Following this thought, we assume that a moving object is an *endurant*. For the scope of this paper, we exclude non-isometric destruction and construction histories, and therefore require the existence of moving objects to last forever by AXIOM 2. *Start(h)* and *finish(h)* of a movable history then are also faces of other connected histories and the velocities at these states always exist and can be zero. Let us call differentiable non-free histories *movable(h)*:

Definition 5. $Movable(h) \Leftrightarrow Differentiable(h) \wedge \neg Free(h).$

It follows a quite obvious theorem for movable subepisodes:

Theorem 1. $Movable(g) \wedge Episode(h,g) \Rightarrow Movable(h).$

We write h⊗h' for the connection of two movable histories, with *h* and *h'* being connectable episodes:

Definition 6. $h \otimes h' := h \cup h'$ *iff* $h'@begin=h@end \wedge Movable(h) \wedge Movable(h').$

We say that a movable history is *larger than* another movable history, $h < superh$, iff there is a connection $h'' \otimes (h \otimes h') = superh$. Now we state an *axiom of infinite existence of movable objects*:

Axiom 2. $Movable(h) \Rightarrow \exists h'. Movable(h') \wedge h < h'.$

Now we are able to define *movements*. A movement is a movable history without zero velocity, which means that it is continuously changing its location from the beginning to the end of its history:

Definition 7. $Movement(h) \Leftrightarrow Movable(h) \wedge \neg\exists t \in]begin(h);end(h)[.\ v_{Traj}(t) = \mathbf{0}.$

Note that the so called *Brownian motion*, as it is nowhere differentiable, and many other kinds of motions, e.g. simulated motions on a network [13], would not be considered a movement.

Using THEOREM 1 it follows that the connected histories of a larger movable history are also movable. Then it is clear that those episodes could be rectangular as well as movements. Now let *h* be a movement history and *h'* be a connected rectangular history. It follows from differentiability of $h \otimes h'$, that the velocities of *h* must ultimately *converge to zero*. In general we can always distinguish between movable histories that converge to zero, and therefore must have a connected rectangular history, and histories which do not, and therefore must have a connected movement history. We call the first ones *bounded movable histories* and the second ones *unbounded*:

Definition 8. $Unboundedmovable(h) \Leftrightarrow Movable(h) \wedge t \in]\ begin(h);\ end(h)[\ \wedge \neg \lim_{t \to end(h)} v_{Traj}(t) = \mathbf{0} \wedge \neg \lim_{t \to begin(h)} v_{Traj}(t) = \mathbf{0}.$

Clearly, unbounded movable histories imply connected movements. This follows from AXIOM 2, DEFINITION 8 and everything that was said above:

Theorem 2. $Unboundedmovable(h) \Rightarrow \exists h'.\ h'@begin=h@end \wedge Movement(h') \wedge \exists h''.\ h@begin=h''@end \wedge Movement(h'').$

We account for motions that can occur on a geographical scale, which we call *friction dominated*. These cannot be modeled in general. What can be modeled are aspects which exist due to the fact that the movement is *constrained* by its support infrastructure, which is the street surface.

4.2 Supports and Channels

Now we clarify the notion of *a movement constraint*. In doing so, we are mostly concerned with rectangular histories *h*, and for these, we write *h* also as an abbreviation for *where(h)*.

Friction dominated movements have *supports*. In terms of Newtonian physics, a support is a physical object whose spatial geometry is such that it provides a normal force for the contact force acting on the supported object. We define *a support* as a solid rectangular regular history with its 3-dimensional spatial extension in *G* having a 2-D face (called a *plateau*) with a vertical component of its surface normal in every point. We therefore assume that the plateau of the support is a *differentiable surface*. Hayes' notion of a *container* is an example of a non-flat support [4]. If a *flat support is* projected into the geographic plane (denoted by the function *plane*), it is identical to the plane-projection of the plateau, so *the plateau covers the cell.*

Definition 9. $Flatsupport(sp) \Leftrightarrow RegularHistory(sp) \wedge Rectangular(sp) \wedge \neg Free(sp) \wedge \exists pl.\ plateau(sp)=pl \wedge FaceOf(pl, sp) \wedge DifferentiableSurface(pl) \wedge \forall s \in pl.\ \exists v. Verticalsurfacenormal(pl, s)=v \wedge \exists i.\ inside(sp)=i \wedge RegularHistory(i) \wedge Rectangular(sp) \wedge Free(i) \wedge FaceOf(pl, i) \wedge plane(pl) = plane(sp) = plane(i).$

For every support, we assume that there is a free space immediately above the plateau of a support, such that the plateau is a face of it and it is equal to the support's plane-projection. We call this space *inside(sp)*. Unlike the inside of Hayes' open container, the height of *inside(sp)* is dependent on the height of the supported objects and is not determined by the support geometry.

For the definition of channels, we think of a *portal as a free, differentiable face of the support's inside* touching the plateau such that its plane-projection is *a face of the plane-projection of the support* (we say that the portal is *on* the boundary of the plateau):

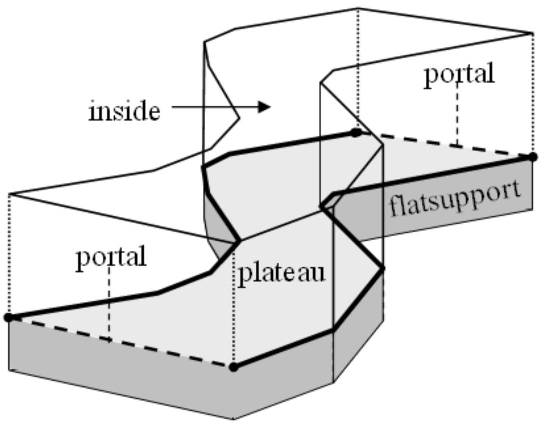


Fig. 5. An illustration of a flat support with two portals and two borders

Definition 10. $Portalof(p,sp) \Leftrightarrow Free(p) \wedge DifferentiableSurface(p) \wedge Flatsupport(sp) \wedge FaceOf(p, inside(sp)) \wedge FaceOf(plane(p), plane(sp)) \wedge \exists c.\ FaceOf(c, p) \wedge FaceOf(c, plateau(sp))$.

In particular, we allow the formation of joined collections of free portals including their common free faces. These objects are again free surfaces. We abbreviate this operation by $\cup A$ for a set of histories $A = \{B, C, D\}$:

Definition 11. $Portal(p) \Leftrightarrow Surface(\cup p) \wedge Free(\cup p) \wedge \forall p_i \in p.\ (\exists sp.\ Portalof(p,sp))$.

A movement is said to *non-tangentially intersect* a free differentiable surface in a point, iff its path intersects it in just that point, and the movement velocity vector does not converge to a vector lying in the tangential plane of the surface at that point:

Definition 12. $Nti(h, p, s) \Leftrightarrow Movable(h) \wedge DifferentiableSurface(p) \wedge Free(p) \wedge \exists s \in G.(path(h) \cap p = s \wedge \exists t, t_i \in when(h).\ Traj^h(t) = s \wedge lim_{ti \to t} v_h(t) = v_{ht} \wedge (s + v_{ht}) \notin tangentialplane(p, s))$.

Note that from this definition it follows that the velocity vector cannot converge to zero (because then it would be in the tangential plane) and the point cannot lie at the non-differentiable boundary of the surface. A movement that non-tangentially intersects two surfaces in its start and finish therefore is unbounded. If a movement non-tangentially intersects the surface in an *inner* point of its trajectory, it is said to *pass* that surface:

Definition 13. $Passing(h, p) \Leftrightarrow \exists s \in G.\ Nti(h,p,s) \wedge s \notin start(Traj^h) \wedge s \notin finish(Traj^h)$.

We state that a movement passing a portal has two connected episodes with one of them having a path being *toso* of the portal:

Theorem 3. $Passing(h, p) \wedge Portalof(p, sp) \Rightarrow \exists e, e'.\ e \otimes e' = h \wedge path(e')^\circ \subseteq toso(p, inside(sp))$.

A proof of this theorem is not presented here, but because the curve does not intersect the surface in one of its end points, it can be divided into two connected parts. Then the only way that two parts of a differentiable curve can stay in the inside is to intersect it tangentially.

We furthermore restrict *supported movement histories* to always be *in contact with a flat support*, so we exclude flying movements, and having their path interior inside of the support's inside:

Definition 14. $Supportedby(h, sp) \Leftrightarrow Flatsupport(sp) \wedge Movable(h) \wedge \forall t \in when(h).\ Incontact(h@t, where(sp)) \wedge path(h)^\circ \subseteq inside(sp)$.

We assume that a supported movable object *has to stay supported* for all its existence, so all its history connections have to stay in contact with a support.

Definition 15. $Supportedmovable(h) \Leftrightarrow \exists sp.\ Supportedby(h,sp) \wedge \forall h'.\ h<h' \Rightarrow Supportedmovable(h')$.

In particular, for a given supported movable history h and its support sp, there always exists a unique *largest supported history of h with respect to sp, h < largestsupported (h, sp).*

Now we can state that an unbounded supported movable history that non-tangentially intersects a portal of its support implies the existence of a flat-joined support with a common portal. We call this theorem the *theorem of unbounded movement support*:

Theorem 4. $\mathit{Supportedmovable}(h) \wedge \exists sp.\mathit{Supportedby}(h,sp) \wedge \exists p.(\mathit{Portalof}(p, sp) \wedge \exists s \in p.\mathit{Nti}(h,p,s) \Rightarrow \exists h', sp', p'.\ \mathit{Movement}(h') \wedge \mathit{Supportedby}(h \otimes h', sp \cup sp')) \wedge \mathit{Portalof}(p', sp') \wedge s \in p' \wedge \mathit{Passing}(h \otimes h', p)$.

Proof. As the movement is supported, it is contained in a support's inside (by DEFINITIONS 14 and 15). Although it non-tangentially intersects a portal, it cannot pass that portal by THEOREM 3, because the portal is on the boundary, and the other side of inside(sp) must lie in its exterior. By DEFINITION 14, it therefore must intersect the portal in either its start or finish. Such a movement must be unbounded (see DEFINITION 13). As the movement is unbounded, by THEOREM 2, there must be a connected following (or preceding) movement. The connection of these movements must be passing the portal by DEFINITION 14, because the intersection point must be an inner point of the movement and it intersects the portal non-tangentially. Then, by THEOREM 3, the connected movement episode is to the other side of the portal. By DEFINITION 15, the connections of the movement must be supported by a second support. As support histories cannot overlap (by AXIOM 1) and the movement is continuous (by DEFINITION 5), there must be a flat-joined history supporting the connected movement. And as the boundary of this support can only be passed at a portal, there must be a common portal that includes the point of non-tangential intersection.

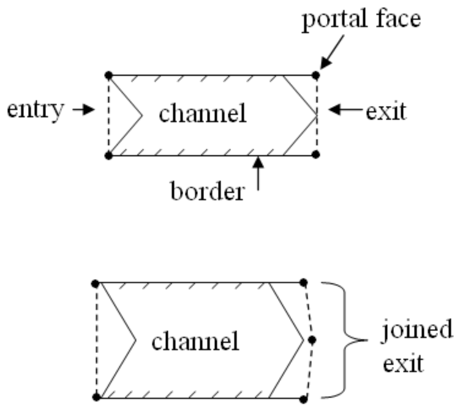


Fig. 6. Illustration (plane projection) of a channel. Exit and entry are portals, so they can consist of a chain of joined free differentiable surfaces.

Now we can think of a *channel* as a kind of flat support with entry and exit portals. We say that channels require their supported movements (or their larger movable histories) to actually *leave at an exit* portal and *enter at an entry* portal:

Definition 16. *Channel(sp)* $\Leftrightarrow$ *Flatsupport(sp)* $\wedge$ $\exists!$*entry, exit. Portal(entry)* $\wedge$ *Portal(exit)* $\wedge$ $\forall p_i \in$*entry.* $\forall p_j \in$*exit. Portalof*(p_i,sp) $\wedge$ *Portalof*(p_i,sp) $\wedge$(*Disjoint*(p_i, p_j) $\vee$ *Joined*(p_i, p_j)) $\wedge$ $\forall h$.(*Movement(h)* $\wedge$ *Supportedby(h,sp)* $\wedge$ *m =largestsupported(h, sp)* $\Rightarrow$ (($\cup$*exit*)$\cap$*path(m)* = *finish*$(Traj^m)$ $\wedge$ $\exists p \in$*exit.Nti(m,p, finish*$(Traj^m)$))) $\wedge$ (($\cup$*entry*)$\cap$*path(m)* = *start*$(Traj^m)$ $\wedge$ $\exists p \in$*entry.Nti(m,p,start*$(Traj^m)$))).

5 Channel Networks as Affordances

Supported movable histories (DEFINITION 15) are the *action* type (compare section 2) of our affordance-based theory, whereas *flat supports* (DEFINITION 9) are the *structure* type. We derive necessary affordance conditions for a network by acting on the *closed world assumption* that all flat supports are part of one single network:

Axiom 3. $\exists!$ *Net. ChannelNetwork(Net)* $\wedge$ $\forall$*SubNet. Flatsupport(SubNet)* $\Rightarrow$ *SubNet* $\subseteq$ $\cup$*Net.*

A *channel network in terms of an action type* can now be defined as a set of channels which must afford a movable history to connect every pair of points on their plateaus:

Definition 17. *ChannelNetwork(Net)* $\Leftrightarrow$ $\forall c \in$*Net.Channel(c)*$\wedge$ $\forall s_1,s_2 \in \cup_{c \in Net}$ *plateau(c).* $\exists$ *h. Supportedmovable(h)* $\wedge$ *plane(start*$(Traj^h)$*)=plane*(s_1) $\wedge$ *plane (finish*$(Traj^h)$*)= plane*(s_2).

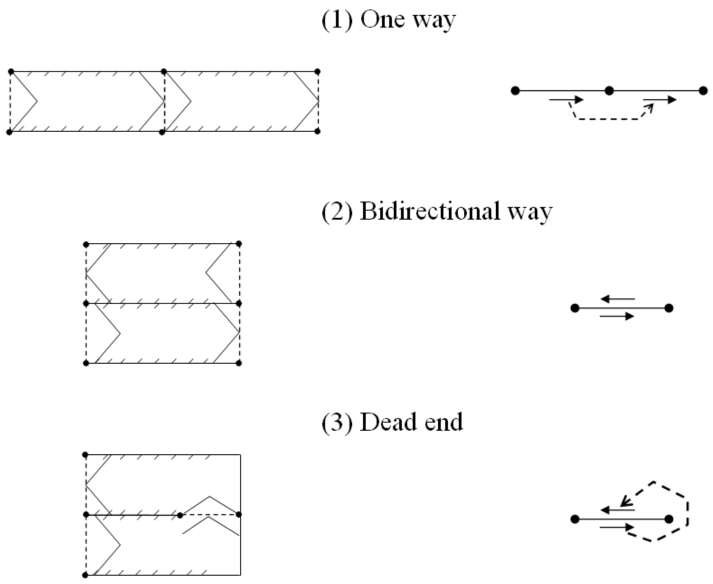


Fig. 7. Some important flat-joined configurations of channels (left) forming road features, and their translations in the graph data model (right) (dotted arrows mean navigation tuples)

From DEFINITION 17 and the closed world assumption many plausible network theorems follow. In particular, it follows that *every channel of a network must be connected to other channels at every one of its portals*. Otherwise the network would exclude movements starting or ending at these portals because of DEFINITION 16 and THEOREM 4. This restricts the possibilities for channel configurations.

We illustrated possible flat-joined channel configurations in FIGURES 7, 8, 9, and one can see that well known road network feature types can be derived by flat-joining channels so as to form common portals. In this way, three-way intersections like bifurcations can be formed (FIGURE 8), as well as 4-way intersections (FIGURE 9).

It is astonishing that a certain type of configuration logically requires also the existence of so called *periodic channels* (see FIGURE 8). These are channels that must alternately cease to exist, because otherwise two logically necessary channels would intersect "at grade", which is prevented by AXIOM 1. In these cases we can presume the existence of a traffic regulator, like a traffic light.

Furthermore, we can obviously distinguish between channels that are connected to the same portal (see number 5 in FIGURE 8 and FIGURE 9), and ones that are connected to two different portals, which we will call *inner* and *outer channels*, respectively. We suspect also that it is provable in our theory that periodic channels are always inner channels.

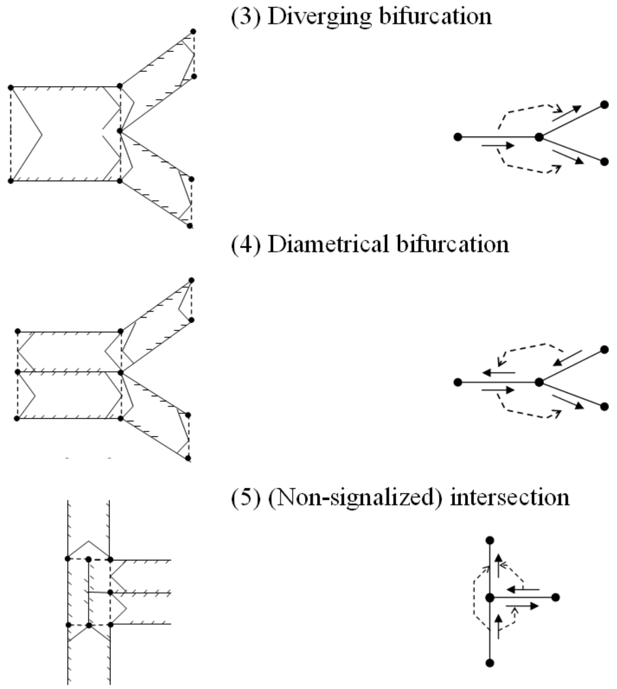

(3) Diverging bifurcation
(4) Diametrical bifurcation
(5) (Non-signalized) intersection

Fig. 8. Flat joined configurations of channels that form 3-valued vertices in the graph data model

On the right hand side of each figure, we included equivalent sub-graphs of the graph data model in section 3, *translated into our theory by the following rules*:

1. Let the set of network nodes *N* in the graph *S* be the set of portals of a network.
2. Let every directed edge of *S* in *L* be an *outer channel* of this network, such that the first vertex is the portal including the channel's entry and the second one is a different portal including the channel's exit. For two bidirectional edges in *S*, let their outer channels be joined at their border.
3. Let a navigation tuple in E mean that its two vertices, translated to *outer channels,* are either connected at a common portal or at a common inner channel.

Let us denote the translations into our theory $\tau(exp)$, with *exp* being a type (e.g. *N*) or a term (e.g. $n \in N$) of the graph data model. We now have an exact notion of the *incompleteness of the data model*: As portals are reduced to a vertex in the graph data model, the information about inner channels and periodic channels (the existence of traffic regulators) is lost. The portals of a channel also include its possible turn off directions, but these are modeled as edges of the navigation relation *E*.

If the translated data model of the set of edges *L*, $\tau(L)$, is a channel network, then the translation has to comply with the following affordance theorems:

Because of AXIOM 3 and DEFINITION 17, *there cannot be other supports than channels of one single network*. Clearly, as $\tau(S)$ consists of channels and portals of one network, there are no other supports existent in the data model.

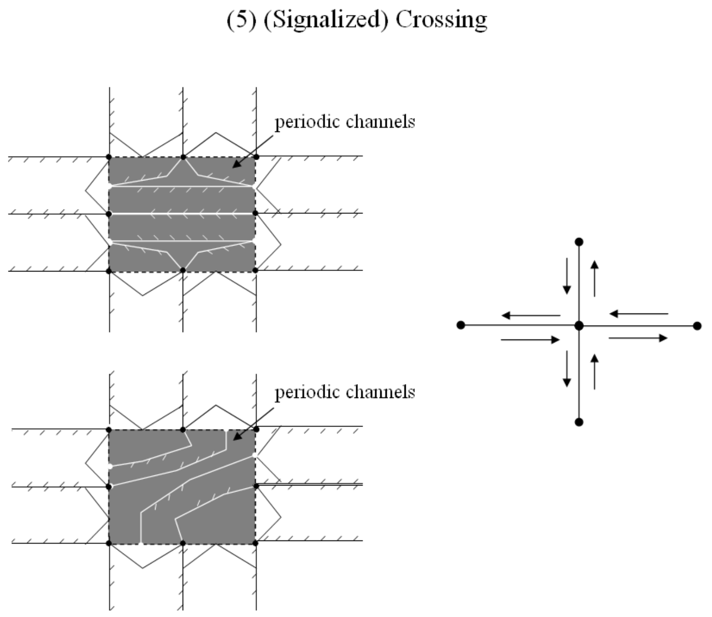


Fig. 9. Flat joined configurations of channels that form 4-valued vertices in the graph data model. These require the existence of periodic channels (signal period), because all channels meet at-grade

Furthermore, *every channel of the network must be reachable from every other channel by a chain of navigable connected channels*. It follows from AXIOM 3 that a movement is supported by a chain of connected channels. According to DEFINITION 17, there must exist a supported movement from every channel to every other one, and so there must also exist an appropriate chain of connected channels, q.e.d.. Now, is this theorem satisfied by the translated data model? As the edges of the graph *S* are translated into outer channels, from the *connectedness of S by navigable paths,* it follows that for each pair of outer channels, there must exist a chain of appropriately connected channels between them, because each tuple of *E* is translated into either a pair of connected outer channels or a triple of connected inner and outer channels.

So, both theorems of the channel network are satisfied by the data model, and from these, all other mentioned characteristics of the data model follow, like the non-existence of graveyards and factories and the minimum vertex degree of 2.

6 Conclusion

Every attempt to classify representations of physical objects in a geographical database into categories is challenged by limited definability and informational incompleteness. We propose to use affordance-based logical theories in the sense of Naïve Physics in order to derive a set of necessary properties, called affordance theorems, for every data model that is supposed to represent such a category. On the one hand, the specification of the afforded action, in our example the movement connectivity of a channel network, comes close to the commonsense a-priori knowledge about the category. On the other hand, through the affordance relation, it becomes possible to derive and explain a-priori unknown structural properties of the physical object, in our example graph connectivity, minimal vertex degree and the non-existence of graveyards and factories.

The proposed method needs to be further elaborated: It is necessary to provide a means to make sure that the proposed set of affordance theorems for the partially interpreted types is in fact *complete*, so that it covers all derivable theorems about these types in the proposed theory. Furthermore, the more interesting categories are road and junction types, which could be defined by induced sub-graphs of the network.

References

1. Diestel, R.: Graph Theory, 2nd edn. Graduate Texts in Mathematics, vol. 173. Springer, New York (2000)
2. Gibson, J.J.: The Ecological Approach to Visual Perception. Houghton Mifflin, Boston (1979)
3. Guarino, N.: Formal Ontology and Information Systems. In: Guarino, N. (ed.) Formal Ontology in Information Systems, FOIS 1998, Trento, pp. 3–15 (1998)
4. Hayes, P.J.: Naïve Physics I: Ontology for Liquids. In: Hobbs, J.R., Moore, R.C. (eds.) Formal Theories of the Commonsense World. Ablex Series in Artificial Intelligence, Norwood (1988)
5. Hayes, P.J.: The Second Naive Physics Manifesto. In: Hobbs, J.R., Moore, R.C. (eds.) Formal Theories of the Commonsense World, Norwood. Ablex Series in Artificial Intelligence (1988)

6. Israel, D.: A Short Companion to the Naïve Physics Manifesto. In: Hobbs, J.R., Moore, R.C. (eds.) Formal Theories of the Commonsense World, Ablex Series in Artificial Intelligence, Norwood (1988)
7. Jordan, T., Raubal, M., Gartrell, B., Egenhofer, M.: An Affordance-Based Model of Place in GIS. In: Poiker, T., Chrisman, N. (eds.) 8th Int. Symposium on Spatial Data Handling, SDH 1998, Vancouver, pp. 98–109 (1998)
8. Kuhn, W.: An Image-Schematic Account of Spatial Categories. In: Winter, S., Duckham, M., Kulik, L., Kuipers, B. (eds.) COSIT 2007. LNCS, vol. 4736, pp. 152–168. Springer, Heidelberg (2007)
9. Lee, J.M.: Introduction to Topological Manifolds. Graduate Texts in Mathematics 202. Springer, New York (2000)
10. Scheider, S., Schulz, D.: Specifying Essential Features of Street Networks. In: Winter, S., Duckham, M., Kulik, L., Kuipers, B. (eds.) COSIT 2007. LNCS, vol. 4736, pp. 169–185. Springer, Heidelberg (2007)
11. Sowa, J.: Knowledge Representation. Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing (1999)
12. Raubal, M., Worboys, M.: A Formal Model of the Process of Wayfinding in Built Environments. In: Freksa, C., Mark, D. (eds.) COSIT 1999. LNCS, vol. 1661, pp. 381–399. Springer, Heidelberg (1999)
13. Van de Weghe, N., Cohn, A.G., Bogaert, P., De Maeyer, P.: Representation of Moving Objects along a Road Network. In: Proc. 12th Int. Conf. on Geoinformatics, pp. 187–194. University of Gävle (2004)
14. Winter, S.: Route specifications with a linear dual graph. In: Richardson, D. (ed.) Advances in Spatial Data Handling: Proc. 10th Int. Symp. Spatial Data Handling (SDH 2002), pp. 329–338. Springer, Heidelberg (2002)

Appendix

A subset of R^m is *bounded* if it is contained in an m-ball of finite radius. A subset of R^m is *connected*, if it cannot be partitioned into two disjoint nonempty closed sets. A *topological n-manifold (with boundary)* (compare [9]) is a Hausdorff space in which every point has a neighborhood homeomorphic to an open subset of Euclidean half-space:

$$\mathbb{R}^n_+ = \{(x_1,\ldots,x_n) \in \mathbb{R}^n : x_n \geq 0\}. \tag{1}$$

Let M be a n-manifold with boundary. The *interior of M*, denoted M°, of dimensionality n, is the set of points in M which have neighborhoods homeomorphic to an open subset of R^n. The n-1 dimensional *boundary of M*, denoted ∂M, is the complement of M° in M. The boundary points can be characterized as those points on the boundary hyperplane ($x_n = 0$) of R^n_+ under some homeomorphism.

A *history* is a subset $h \subset R^4$ of 4-dimensional space-time, with the first three real valued dimensions denoting geographical space $G := R^3$, and the fourth real valued dimension denoting time $T := R$, having the following properties:

1. h is a *topological n-manifold (with boundary)*, with $1 \leq n \leq 4$.
2. the projection of h into T (denoted by when(h)) is a *non-degenerate interval*, so it is connected and neither the empty nor the singleton set. We say that h is "temporally extended".

3. each projection of h into G for any moment $t \in T$ (denoted by h@t) is a *bounded and connected* subset of G, so a history is said to be "spatially bounded" and "one piece".

Every *closed m-1 manifold embedded in R^m* (e.g. a closed 2-sphere in R^3) divides the space R^m into two unique disjoint open subspaces, one of which is the bounded *interior* and the other one being the unbounded *exterior.* As this is always the case for the boundary of an m-manifold (e.g. a 3-ball), for every boundary part *f* (called "*face*") of this m-manifold, the function *toso(f,e)* is a bijective mapping between the two subspaces.

A Theory of Change for Attributed Spatial Entities

John Stell[1] and Michael Worboys[2]

[1] School of Computing, University of Leeds, Leeds, LS2 9JT, UK
[2] Department of Spatial Information Science and Engineering, University of Maine, Orono, ME 04473, USA

Abstract. New methods of data collection, in particular the wide range of sensors and sensor networks that are being constructed, with the ability to collect real-time data streams, provide a driver for an appropriate underlying theory for information related to dynamic geographic phenomena. This paper investigates the underlying processes by which entities with both spatial and aspatial components may evolve and change through time, and how the spatial and aspatial dimensions participate separately and together in such an evolution. The overall structure that we propose is that of attributed locations in space-time. After motivation and a survey of relevant background work, the paper introduces a formal framework and presents a case study that applies it to a detailed example. The focus is the impact of spatial change on attribute change, but also considered is the converse process. We conclude by discussing the relevance of this work to the extraction of dynamic objects and their changes from sequences of temporal snapshots of static scenes.

1 Introduction

New methods of data collection, in particular the wide range of sensors and sensor networks that are being constructed, with the ability to collect real-time data streams, provide a driver for an appropriate underlying theory for information related to dynamic geographic phenomena. There has already been research, some of which is detailed in section 2, into the purely geometrical and topological aspects of such phenomena. This paper describes research that also brings into play some of the non-spatial aspects, sometimes called the attribute or semantic components of geographic information. We investigate the underlying processes by which entities with both spatial and aspatial components may evolve and change through time, and how the spatial and aspatial dimensions participate separately and together in such an evolution.

Components of a geospatial feature include not only spatial properties (e.g., location, shape, size) and relationships (Euclidean, metric and topological relations), but also attribute properties (e.g., level of a scalar value such as temperature). In a collection of features, objects may change in their spatial relationships with each other in addition to changes to themselves. Objects may come in to the scene, as well as leave it. The events themselves may be part of larger patterns, comprising of many "atomic events." A sequence of snapshots is insufficient in itself, without further reasoning, to determine the underlying changes.

To take a very simple example, the left hand side of figure 1 shows two snapshots, A and B, of the evolution of a region. What cannot be determined is whether the change

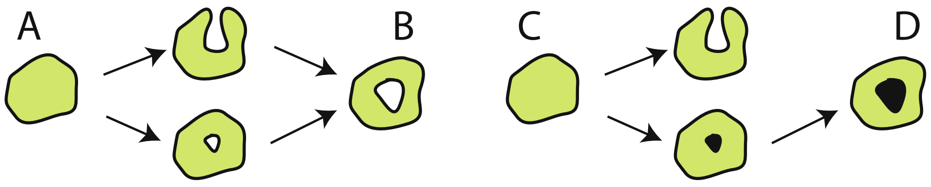


Fig. 1. Engulfment or hole growth?

from A to B results in engulfment, as in the upper intermediate state, or creation of a hole, as in the lower. It is interesting that information about non-spatial properties of the entity (in this case, color), provides the possibility of more finely-tuned reasoning. Looking at the right-hand part of figure 1, if we assume that colors of individual regions cannot change, except by growth of new components, then the engulfment process can be eliminated from the evolution of the region from C to D. Although this is an artificial and simplified example, it shows the framework presented in this paper, namely analysis of the evolution of mixes of spatial and aspatial properties.

Figure 2 shows a typical representation of spatial data, taken from a sample of the Ordnance Survey of Great Britain's MasterMap dataset. In it we see the spatial footprints of real world entities represented by polygons, while other attributes are indicated by text and color. When we look more closely at the representation we see a mix of data about particular entities, such as 'Manchester House' and the polygon that represents its spatial footprint, as well as more general information, such as 'PH', indicating that an entity whose spatial footprint is represented by a specific polygon is in the category of public houses, or the color blue indicating a road. Now, this representation is purely static, involving neither spatial nor aspatial change. If such changes are allowed, a range of questions arises, such as, 'How should a merge of the spatial footprint of one of the public houses merges with a neighboring house be represented?' 'What happens to the representation if Manchester House becomes a public house?'. Now the answer to any one of these questions might be straightforward, but reasoning about change requires a general foundation, and it is this foundation that is presented here.

The overall structure that we propose is that of attributed locations in space-time. The attributes may be natural-language, symbolic, or in any other appropriate form. In the case of the map in figure 2, the attributes are a mix of natural language, symbols, and colors. The choice of form of attribute is partly a question of representation, but more fundamental is the issue of the different ontological categories that the labels indicate. One dichotomy is between the *universal* and the *particular*. Universals have instances, are repeatable, abstract, and lack specific locations in space-time; while particulars have a unique spatio-temporal location. In figure 2, 'public house' is a universal while 'Manchester House' is a particular. Universals are commonly divided into *types*, *properties*, and *relations*. Types may be thought of as having instances that are 'objects', in some sense. Thus 'public house' is a type, whose instances are particular public houses at specific locations. An example of a property is 'rectangular', referring to the shape of a land parcel, and an example of a relation is 'adjacent', indicating a relationship between two land parcels. We will see in what follows that it is important

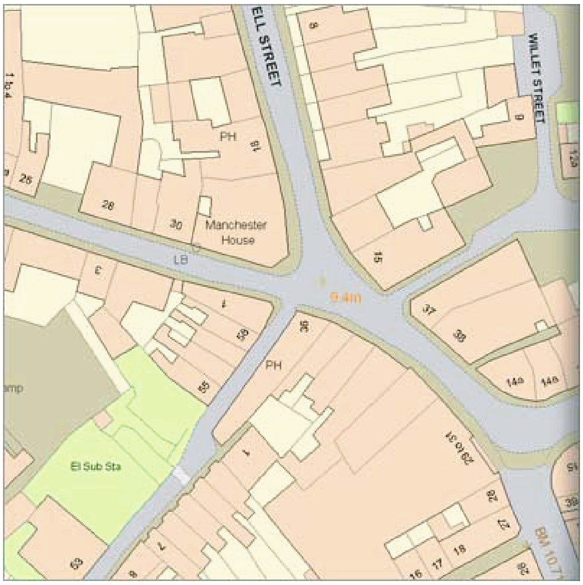


Fig. 2. Typical representation of spatial data (Reproduced by permission of Ordnance Survey. (c) Crown Copyright).

to know what category our attributes fall into, as well as the structure of the category, in order to perform a correct analysis.

This paper reports research on the interaction between instances and their types as they evolve. The evolution will often be considered to be through time. So, an example of a typical question is, when two patches of liquid attributed by distinct liquid types become merged, what is the type associated with the merged instance? But the framework also applies when the evolution is in representation, as for example when a spatial representation is generalized. For example, suppose that a land parcel attributed with type house is merged with a land parcel of type garden, what is the type of the merged parcel? Our framework will be seen to cover both these cases.

The structure of this paper is as follows. After this introduction is given a review of some relevant background work. We will then introduce the framework and present a case study that applies it to a detailed example. The next section considers related questions, especially the dual issue of the impact of a change of types upon the representation of the instances. The paper concludes with a summary and consideration of future directions.

2 Background

Research in the area of spatio-temporal information systems is now well-established. Early work includes a general survey of early work with a database orientation [1]; a description of a proposal for a temporal query language [9], and construction of a model of spatio-tempral information with a focus on region evolution [13]. An important event that established a research agenda for spatio-temporal reasoning in geographic spaces was the 1993 National Center for Geographic Information and Analysis Specialist Meeting [4].

Analysis of dynamic scenes, particularly as found in the computer vision community, has traditionally been quantitative and data-intensive (see, for example, [5]). General foundational work on types of object evolution, not specifically related to spatial change is reported by Hornsby and Egenhofer in [7]. The foundation for giving first-class status to events was set out ontologically by Grenon and Smith in [6] and from an information modeling and reasoning perspective by Worboys in [14]. Grenon and Smith extended the usual ontologies of things in snapshot (so-called SNAP ontologies) to include entities that are changes, occurrences, processes, and events (SPAN ontologies). Worboys analyzed changing spatial entities, including dynamic spatial phenomena from the SPAN perspective, and showed how techniques applied to computational processes, including process calculi and the event calculus used in artificial intelligence for robotic reasoning, could be applied in the geospatial context. The specifically spatial dimension of change has been analyzed in the context of basic topological change by Jiang and Worboys in [8]. This builds upon work on transitions between topological states, as found for example in [3], where a network structure is presented in which node represent topological relationships between two regions, and two nodes are direct neighbors if no sudden spatial "jump" is needed to get from one relationship to the other. Stell [11] considered how evolving entities could be described at different levels of detail through a granular account of time.

Some of the inspiration for the formal structures developed later in the chapter comes from work on Chu categories as developed in theories of information transfer between components of distributed systems. A good reference for this work is [2] on information flow and [12] on the more formal properties of Chu spaces.

There are many applications of spatial change reported in the literature, from the engulfment of cytoplasm by bacteria, to patterns of evolution of a gene sequence, to land type and use change. Research on evolutionary biology has also inspired some of the idea in this proposal, in particular work by theoretical biologists on topological spaces in patterns of evolutionary change (see, for example, [10]).

3 Formal Framework

In this section we set out the formal properties that we expect the operation of combination to possess. We then outline the basic formal structure, around which this work is based, and show how it applies to a simple example. Assume that we have a set of spatial objects (e.g., regions, or links on a network), each of which has some attribution. For example, a region of land might have a land type associated with it. We make the simplifying assumption that each object has one and only one attribute (although of course, such an attribute might be a vector of simpler types). So, let X be a set of spatial objects and T be a set of attributes (called here *types*).

We assume that the spatial objects may change and combine in terms of their spatial extents. Thus, an object might move, rotate, dilate, grow a hole, merge with another,

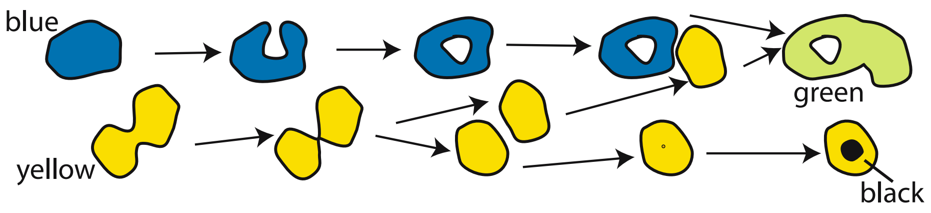


Fig. 3. Splitting and merging attributed spatial objects

split, engulf another, and so on. We will focus specifically on the types of movement involving changes in the topology of the total assemblage. Merging, splitting, insertion, and deletion provide the principal topological changes to the spatial extents under consideration here. The investigation of these topological changes on their own, with no attributes, is the subject of earlier research [8]. Their possible influence on attribute changes is shown by an example in figure 3. Here we see the evolution of a pair of areal objects, attributed in this case by colors. The top object merges with itself, forming a hole, while the lower one splits. One piece of the lower object migrates and merges with the upper object, while the other develops a hole. In the case of colored liquids, we can imagine the effect to be as shown.

3.1 Combining Entities

The focus of this work is the effect of topological change on the attribution of areal objects. To begin to understand this, we must consider the structure of the domain from which the attributes are taken, and the effect of this structure on combination, splitting, and recombination. It is clear that a combination of colors, that might occur in the mixing of paints, is rather different from a combination of land use types that might take place in the generalization of a spatial representation. There are other cases, such as the combination of a husband and wife into a married couple that introduce further complexities. To indicate the differences, we might reasonably say that a husband is part of a married couple, and but maybe less obvious that the color blue is a part of the color green or that the type house is part of the type residence. In the first case, the structure of the domain is mereological, while in the second and third examples, we have a subsumption structure.

We now make some assumptions about the domains of interest, and the properties that mixing or combination has for those domains. Let T be the domain of attributes. For the set T, we impose a structure that models the ability for types to combine. Define a binary operation $\vee$ on T, where $a \vee b$ which is to be interpreted the type formed by the mix or combination of types a and b. We impose the following properties on the structure $\langle T, \vee \rangle$:

- For all $a, b, c \in T$, $(a \vee b) \vee c = a \vee (b \vee c)$. (Associativity)
- For all $a \in T$, $a \vee a = a$. (Idempotence)
- For all $a, b \in T$, $a \vee b = b \vee a$. (Symmetry)
- There is an element $\perp$, such that for all $a \in T$, $a \vee \perp = a$. (Existence of a bottom element)

$$\begin{tikzcd}
 & T & \\
X \arrow[ur, "t"] \arrow[rr, "r"'] & & Y \arrow[ul, dashed, "u"']
\end{tikzcd}
\qquad
\begin{tikzcd}
 & T & \\
PX \arrow[ur, "t^+"] & & PY \arrow[ll, "r^*"] \arrow[ul, dashed, "u^+"']
\end{tikzcd}$$

Fig. 4. Spatial object reformation

These properties can be interpreted in a natural and intuitive way as properties of combination. Combination is assumed to be associative, and idempotence reflects the idea that mixing a quality with itself produces no change. We assume that the order of combination is not important (symmetry), and that there is a "zero" quality that when mixed with any quality leaves it unchanged. This is just one set of plausible properties, and the application would determine what formal properties would form the basis of an appropriate model. In the case above, $\langle T, \vee \rangle$ has the structure of a join semilattice with bottom element. As for the collection X of spatial objects, to begin with we give it no more structure than that of a set.

3.2 Combination of Attributed Objects

The first situation under discussion is represented by the arrow diagram shown in the left of figure 4. The objective is to find the effect on the typing of spatial objects if they are reformed. This shows a reformation of the spatial objects from the set X to the set Y, represented as a relation r from X to Y. So, for example, the merge of two objects x_1 and x_2 into object y is represented by $x_1\,r\,y$ and $x_2\,r\,y$, while the split of object x into objects y_1 and y_2 is represented by $x\,r\,y_1$ and $x\,r\,y_2$. The function t is a typing function, indicate the type tx of each element $x \in X$. We wish to construct typing function u, or at least to determine constraints upon its construction. So for example, we might expect that the type of the merged object y above reflects the types of its constituent objects, and that the types of the objects y_1 and y_2 are derived from the object from which they split. The exact rules will depend upon the application, so here is presented to outline of a general theory. We are assuming in this situation that the collection of types T is unchanged. This will be generalized later.

It is convenient to work wholly with functions rather than a mix of functions and a relation. To do this, we consider collections of objects, rather than the individual objects themselves. Hence, we construct functions $r^* : PY \to PX$ and $t^+ : PX \to T$ as follows.

$$r^* : B \mapsto \{x \in X \mid x\,r\,b \text{ for } b \in B\}, \text{ where } B \subseteq Y \tag{1}$$

$$t^+ : A \mapsto \bigvee_{s \in A} ts, \text{ where } A \subseteq X \tag{2}$$

It is not difficult to see that if we can find function $u^+ : PY \to T$, it is then possible to reconstruct function u by the rule that $uy = u^+\{y\}$. All that remains is to state the

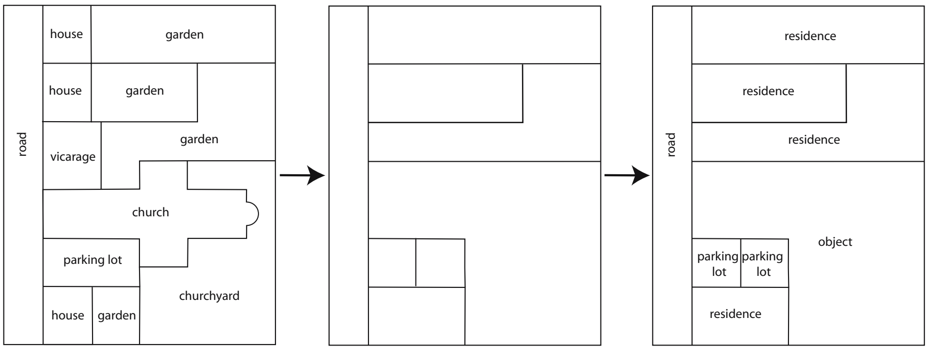


Fig. 5. Spatial scene reformation

rule for constructing function u^+, (alternatively, the constraint on u^+). This is given by equation 3.

$$u^+ = t^+ r^* \tag{3}$$

3.3 Example: Spatial Representation Generalization

Let us work through this formalism by means of an example. The left hand portion of figure 5 shows a configuration of spatial objects. This configuration can be represented as a pairing of a set of instances $X = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}, \mathsf{g}, \mathsf{h}, \mathsf{i}, \mathsf{j}, \mathsf{k}, \mathsf{l}\}$ with a set of types T. The types have the additional structure of a join semilattice, and are shown in figure 6. The explicit pairings are shown in figure 7, where in this case each instance in X is related to a unique type in T through its spatial situation as an areal object. Because the relationship between instances and spatial situations is a 1-1 correspondence, the pairing t between instances and types is functional. So, for example $t : \mathsf{a} \mapsto \mathsf{house}$.

Now, imagine that a map reformation operation entails the merging and splitting of areal objects, as shown in the middle portion of figure 5. This reformation entails a relation r from instances to merged/split instances, as shown in figure 8. For example, the regions attributed a and b have been merged into the region attributed a', and the region attributed i has been split into the regions attributed f' and g'. Relation r is fully specified by, $\mathsf{a}r\mathsf{a}'$, $\mathsf{b}r\mathsf{a}'$, $\mathsf{c}r\mathsf{b}'$, $\mathsf{d}r\mathsf{b}'$, $\mathsf{e}r\mathsf{c}'$, $\mathsf{f}r\mathsf{c}'$, $\mathsf{g}r\mathsf{d}'$, $\mathsf{h}r\mathsf{e}'$, $\mathsf{i}r\mathsf{f}'$, $\mathsf{i}r\mathsf{g}'$, $\mathsf{j}r\mathsf{e}'$, $\mathsf{k}r\mathsf{h}'$, and $\mathsf{l}r\mathsf{h}'$.

The function r^* defined by equation 1 can now be constructed. Firstly, construct the action of r^* on singleton sets, with, for example:

$$\begin{aligned} r^* &: \{\mathsf{a}'\} \mapsto \{\mathsf{a}, \mathsf{b}\} \\ r^* &: \{\mathsf{b}'\} \mapsto \{\mathsf{c}, \mathsf{d}\} \\ r^* &: \{\mathsf{f}'\} \mapsto \{\mathsf{i}\} \\ r^* &: \{\mathsf{g}'\} \mapsto \{\mathsf{i}\} \end{aligned}$$

Then, the action of r^* on non-singleton subsets is computed by taking the union of its actions on the singleton constituents. So, for example:

$$r^* : \{\mathsf{a}', \mathsf{b}'\} \mapsto \{\mathsf{a}, \mathsf{b}\} \cup \{\mathsf{c}, \mathsf{d}\} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$$

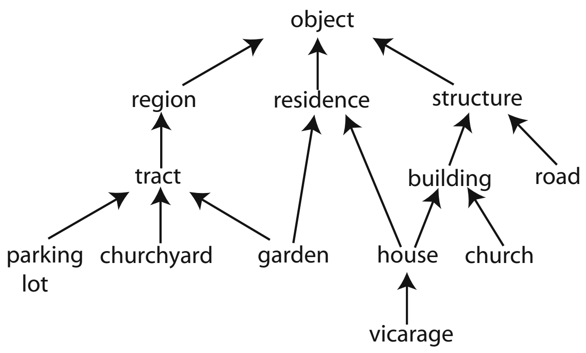


Fig. 6. Land typology

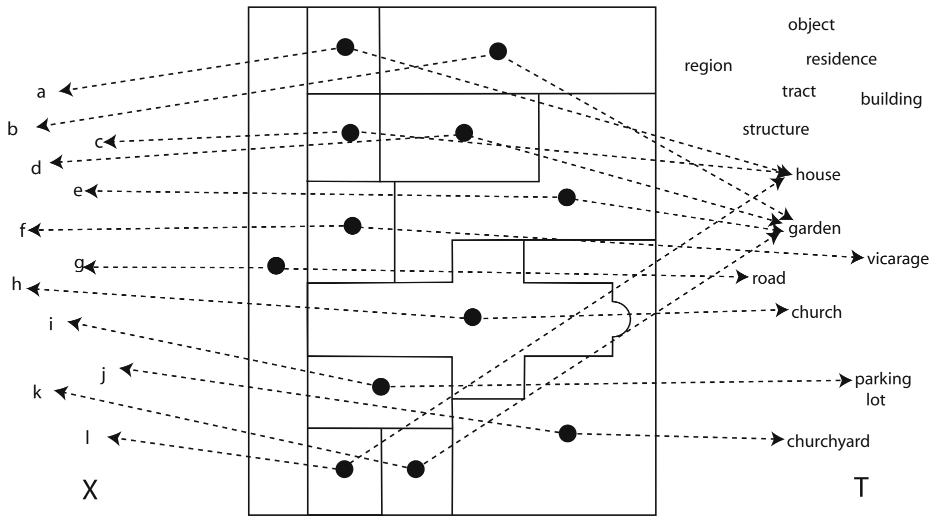


Fig. 7. Links between objects and types

In a similar way, function t^+ defined by equation 2 is constructed by computing its action on singleton sets, with, for example:

$$t^+ : \{\mathsf{a}\} \mapsto \mathsf{house}$$
$$t^+ : \{\mathsf{b}\} \mapsto \mathsf{garden}$$
$$t^+ : \{\mathsf{i}\} \mapsto \mathsf{parking\ lot}$$

Then, the action of t^+ on non-singleton subsets is computed by taking the join of its actions on the singleton constituents. So, for example:

$$r^* : \{\mathsf{a}, \mathsf{b}\} \mapsto \mathsf{house} \vee \mathsf{garden} = \mathsf{residence}$$

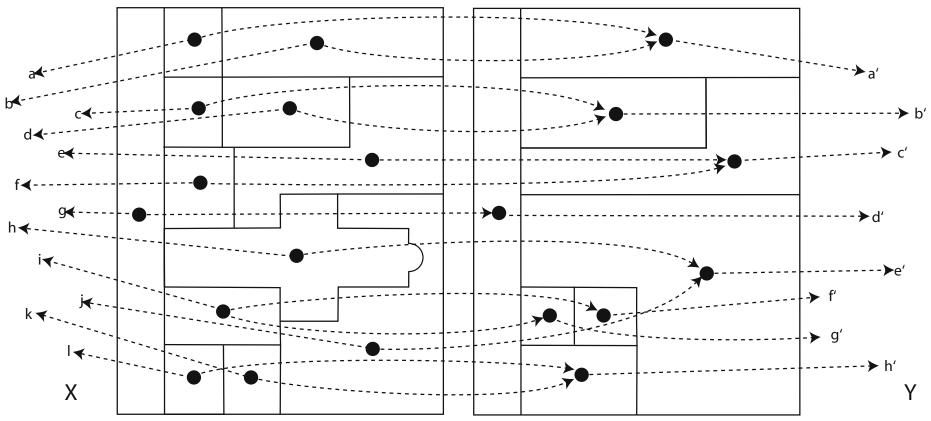


Fig. 8. Links between objects and their splits and merges

The final step is to construct function u^+, as defined by equation 3, as the composition of t^+ with r^*. So, for example:

$$u^+ : \{\mathsf{a}'\} \mapsto \{\mathsf{a}, \mathsf{b}\} \mapsto \textsf{residence}$$
$$u^+ : \{\mathsf{f}'\} \mapsto \{\mathsf{i}\} \mapsto \textsf{parking lot}$$
$$u^+ : \{\mathsf{g}'\} \mapsto \{\mathsf{i}\} \mapsto \textsf{parking lot}$$

This enables us to complete the attributes for each land area, as shown in the right-hand panel of figure 5. The example computations above are shown for the cases of a merge and a split, and in both cases lead to the result we would expect.

4 Further Investigations

The preceding section set out the underlying structure, and showed how reformation of the scene of spatial objects implies a reconfiguration of types. In this section we look at one extension of this idea, where the types may also evolve, and then consider the converse case where a change of types will effect a reformation of the spatial scene.

4.1 Evolution of the Type Structure

In this section, we not only allow the spatial objects to change, but also the types. Changes to the type domain correspond in database terminology to *schema evolution*. This more general framework is shown in figure 9. There are two type domains, T and U, and a function s between them. In some cases, it might be appropriate to constrain s to be a function that preserves the structure of the type domains, so in our paper s would be a join-semilattice morphism. For example in the case of spatial representation generalization, it might well be appropriate to assume that the type domain for the generalized map is a homomorphic image of the source type domain. However, this is not essential in what follows.

$$\begin{tikzcd}
T \arrow[r, "s"] & U \\
PX \arrow[u, "t^+"] & PY \arrow[l, "r^*"] \arrow[u, "u^+"']
\end{tikzcd}$$

Fig. 9. Generalized reconfiguration arrow diagram

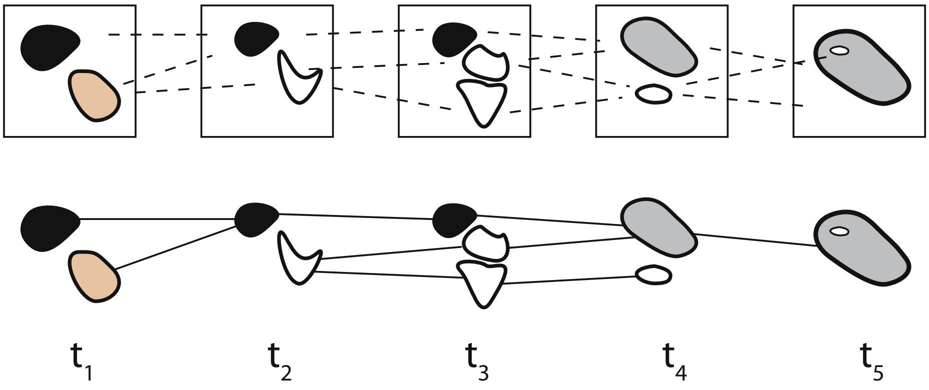


Fig. 10. Extracting dynamic objects from sequences of scene snapshots

The extension is quite straightforward. The equations 1 and 2 are as before, but equation 3 is modified to:

$$u^+ = s\,t^+ r^* \tag{4}$$

4.2 Effect of Type Changes upon the Underlying Spatial Representation

Next, we investigate the degree to which merging attributes has an impact on the spatial representation. In maps, we can observe that a common annotation between neighboring regions might allow cross-border integration. Returning to the example of section 3.3, suppose that we assume some combination of types, and wish to infer a reformation of the spatial representation. For concreteness, assume we have the following mapping $s : T \to U$, given by s : house $\mapsto$ residence, s : garden $\mapsto$ residence, s : vicarage $\mapsto$ residence, and s acts as the identity on all other members of T. (Note by the way that

$s : T \to U$ is a join-semilattice morphism.) If we followed through a process dual to that in section 3.3, we would like to be able to conclude that a pair of spatial regions that are attributed house and garden in T have the possibility of merging into a single region attributed by residence in U. Of course, we do not want all such pairs of objects to be merged. Firstly, they must be share a common boundary, and even that is insufficient (note regions attributed f and d in figure 8. We need more information to correctly merge the appropriate regions. This points to a basic lack of duality between spatial objects and their types.

5 Conclusions

In this paper, we have argued that static representations of the beginning and ending of a change are insufficient in general to determine the nature of the change. The change needs itself to be explicitly represented. To this end, we developed earlier work on the explicit and formal representation of purely spatio-topological change so that changes of attributed spatial objects may be also represented. The formalism was applied to an example where changes take place to a representation through the process of generalization. A key issue was the effect that the structure of the spatial and attribute domains had upon the overall structure, and we examined the particular case where the attribute domain had the structure of a join-semilattice under attribute combination. In the first stage of the work, only the spatial domain was allowed to evolve through change, but in the second we extended the formalism to allow both spatial structure and attribute domain evolution.

One of the as yet only partially solved problems in spatio-temporal reasoning is the extraction from a temporal sequence of spatial snapshots a collection of dynamic and evolving spatial objects. We expect the formalism developed here to allow the construction of constraints imposed by pairs of spatial snapshots to possible object evolutions within the sequence. Figure 10 shows an example of the process carried through five time steps, t_1 up to t_5. The top portion of the figure shows the dynamic spatial scene, comprising of a temporal sequence of scene snapshots. The approach is to utilize the formal constraints, as developed above, to identify possible relationships between the objects in consecutive snapshots. These relationships will be labeled with uncertainty levels, according to weight of evidence that a relationship exists. In the lower half of the figure, the weighted relationships have been used to determine the dynamic object and its change relationships. In the example, there are merge, creation, splitting, attribute change, attribute combination, destruction, and hole formation change events. This is the subject of ongoing work.

Acknowledgments

This material is based upon work supported by the US National Science Foundation under Grant No. IIS-0534429 and the UK Engineering and Physical Sciences Research Council under Grant No. EP/F036019/1.

References

1. Abraham, T., Roddick, J.: Survey of spatio-temporal databases. GeoInformatica 3, 61–69 (1999)
2. Barwise, J., Seligman, J.: Information Flow. Cambridge University Press, Cambridge (1997)
3. Cohn, A.G., Gotts, N.M., Randell, D.A., Cui, Z., Bennett, B., Gooday, J.M.: Exploiting temporal continuity in qualitative spatial calculi. In: Golledge, R.G., Egenhofer, M.J. (eds.) Spatial and Temporal Reasoning in Geographical Information Systems. Elsevier, Amsterdam (1997)
4. Egenhofer, M., Golledge, R. (eds.): Spatial and Temporal Reasoning in Geographic Information Systems. Oxford University Press, Oxford (1998)
5. Fernyhough, J., Cohn, A.G., Hogg, D.C.: Constructing qualitative event models automatically from video input. Image and Vision Computing 18, 81–103 (2000)
6. Grenon, P., Smith, B.: SNAP and SPAN: Towards dynamic spatial ontology. Spatial Cognition and Computation 4(1), 69–104 (2004)
7. Hornsby, K., Egenhofer, M.J.: Identity-based change: A foundation for spatio-temporal knowledge representation. International Journal of Geographical Information Science 14(3), 204–207 (2000)
8. Jiang, J., Worboys, M.F.: Event-based topology for dynamic planar areal objects. International Journal of Geographical Information Science (under review, 2008)
9. Snodgrass, R. (ed.): The TSQL2 Temporal Query Language. Kluwer Academic Publishers, Dordrecht (1995)
10. Stadler, B., Wagner, G., Fontana, W.: The topology of the possible: Formal spaces, underlying patterns of evolutionary change. Journal of Theoretical Biology 213, 241–274 (2001)
11. Stell, J.G.: Granularity in change over time. In: Duckham, M., Goodchild, M.F., Worboys, M.F. (eds.) Foundations of Geographic Information Science, pp. 95–115. Taylor and Francis, London (2003)
12. van Benthem, J.: Information transfer across Chu spaces. Logic Journal of IGPL 8(6), 719–731 (2000)
13. Worboys, M.F.: A unified model of spatial and temporal information. Computer Journal 37(1), 26–34 (1994)
14. Worboys, M.F.: Event-oriented approaches to geographic phenomena. International Journal of Geographic Information Science 19(1), 1–28 (2005)

Delineation of Valleys and Valley Floors

Ralph K. Straumann and Ross S. Purves

Department of Geography, University of Zurich,
Winterthurerstrasse 190, CH-8057 Zurich, Switzerland
{ralph.straumann,ross.purves}@geo.uzh.ch

Abstract. Methods to automatically derive landforms have typically focused on pixel-based, bottom-up approaches and most commonly on the derivation of topographic eminences. In this paper we describe an object-based, top-down algorithm to identify valley floors. The algorithm is based on a region growing approach, seeded by thalwegs with pixels added to the region according to a threshold gradient value. Since such landforms are *fiat* we compare the results of our algorithm for a particular valley with a number of textual sources describing that valley. In a further comparison, we computed a pixel-based six-fold morphometric classification for regions we classified as either being, or not being, valley floor. The regions classified as valley floor are dominated by planar slopes and channels, though the algorithm is robust enough to allow local convexities to be classified as within the valley floor. Future work will explore the delineation of valley sides, and thus complete valleys.

Keywords: Landform, geomorphometry, valley, valley floor, delineation.

1 Introduction

Since the early 1990s the explosion of availability of spatial data in general, and data describing the elevation of the earth's surface in particular, has led to considerable effort by geomorphologists and GIScientists to develop techniques capable of describing and delineating the features which go together to make up a landscape. In many ways, such research is a return to early ideas expressed by Maxwell [1] in his treatise "On hills and dales". In this paper, we are particularly interested in the delineation of landforms, that is to say attempts to answer questions such as "*Where* is a mountain?" posed by Fisher et al. [2]. However, in our exploration of the literature we noted a concentration by GIScientists on the delineation of mountains (e.g. ibid., [3]) or, as they are also more neutrally termed, *topographic eminences* [4]. Further, Hugget [5], a geomorphologist, states that "valleys are so common that geomorphologists seldom defined them and, strangely, tended to overlook them as landforms". Thus, in this paper we describe research which aims to explore a range of techniques for the extraction of valleys, concentrating on *valley floors*. Importantly, we are concerned not with the extraction of channel networks (the *thalweg*), but with deriving the spatial extent associated with a valley floor.

An important prerequisite to extracting landforms is to consider what the term itself implies. In the literature there appear to be two contrasting, and for GIScientists familiar, sets of definitions. The first set of definitions are essentially field-based. For example, Whittow [6] defines landform as "the morphology and character of the land surface that results from the interaction of physical processes (...) and crustal movements with the geology of the surface layers of the Earth's crust". Following this definition, landforms are defined by fields of continuous attributes (e.g. gradient and curvature) and a multitude of what we will term pixel-based extraction techniques have been developed which set out to extract landform elements on the basis of similarities in these attributes. The second set of definitions view landforms as objects, with for example Lapidus et al. [7] and Blasczczinski [8] giving examples of landforms which include mountains, valleys, rivers and canyons.

These two sets of definitions lead, in turn, to two different approaches to the problem of extracting landform(s) (elements). In the case of a field-based view of landforms, the problem is essentially bottom-up. Attributes are defined over an entire landscape, and a range of techniques applied to identify areas within a landscape with similar attribute values. By inspection of attribute values within similar landform elements, they may be assigned either a name reflecting simply these attribute values (e.g. double-convex slopes), or be associated with a landform (e.g. convex local maxima may be assigned to the landform topographic eminence). In an object view of landforms the essential difference is that the starting point is some notion of the landform under investigation, which in turn leads to a top-down method. Thus, our starting point is to characterise a landform of interest (e.g. a valley) before applying a range of methods to delineate the boundaries or, if we adopt a fuzzy approach similar to that proposed by Fisher et al. [2], the *valleyness* of locations.

In this paper we describe a case study with a special focus on Gürbe valley in Switzerland, using techniques based on both popular notions of Gürbe valley and a top-down method developed to extract valley floors from DEMs. We firstly set out a range of related work on the extraction and definition of landforms and landform elements, as well as the application of Naïve Geography to delineating object boundaries and list a series of definitions of the landforms valley and valley floor. We then describe simple methods to extract Gürbe valley from natural language descriptions, before we introduce a DEM-based algorithm for the delineation of valley floors. The algorithm results are illustrated over Switzerland as a whole before we describe them in detail for the Gürbe valley and relate them to the natural language descriptions and compare our results with a pixel-based classification.

2 Related Work

2.1 Describing Landscapes in Terms of Surface Form

Geomorphometry can be defined as the quantitative measurement and analysis of the form of the earth's surface. A range of attributes are used in describing this form (cf. e.g. [9, 10]). In GIScience terms these attributes can be split into focal, zonal and global measures. Basic focal attributes encompass the first order derivatives of elevation, i.e. slope in terms of magnitude (gradient) and direction (aspect) [11, 12]. Profile, plan and a variety of additional curvature measures (cf. [13]) are second order derivatives of elevation. All these measures can be approximated from a moving window through a focal operation, typically with a neighbourhood of 3x3 cells, though Wood [14] and others have emphasised the importance of scale (and thus varying window size) on the derivation of such attributes.

Zonal attributes are computed within some defined analysis region. Local relief [15] defined as the range of elevations in an area is an example. The hypsometric curve and the hypsometric integral are also calculated over a pre-defined region, typically a drainage basin.

Global attributes, which in principle can consider any cell within the study area, are typically more complex to compute. Examples are the distance to a local depression, the elevation above local depression [16] or ridge proximity [17].

Compound derivatives combine two or more terrain attributes which may be focally, zonally or globally defined. Examples in geomorphology include the topographic wetness index [18, 19] and the stream power index [10].

Pixel-Based Bottom-Up Approaches. Pixel-based bottom-up approaches are numerous. They can be roughly subdivided into supervised (classification) and unsupervised (clustering; classical bottom-up) approaches. The latter first choose attributes on which the clustering process is to take place, before forming either crisp or fuzzy clusters by minimising intra-class variance and maximising inter-class variance (e.g. [20, 17, 21]).

Supervised classification utilises either training data or values from the literature to identify clusters within data. Pennock et al. [22] proposed a seven-fold crisp classification based on Ruhe's [23] profile form classes. A similar scheme was put forward by Dikau [24] and modified by Wood [14]. These classification schemes have – sometimes in adapted or extended form – often been applied to derive both crisp [25, 26, 27, 28] and fuzzy [29, 30] classifications. Wood [14] proposed multi-scale classification which was extended into fuzzy multi-scale classification based on a range of crisp classifications at different scales [2].

Object-Based Top-Down Approaches. Besides pixel-based approaches characterisations of topography can result in defined objects, rather than classified pixels (clearly, pixel-based characterisations can be used to derive objects by the identification of some threshold value). For example, Lucieer and Stein [31] use a texture measure, as well as other attributes, to seed a region growing algorithm. However, no *a priori* knowledge about landforms is utilised, making the process essentially data-driven.

Fisher et al. [2] present a partially top-down approach where they reason about the essence of peaks and their relationship to summits. They use fuzzy multi-scale classification into six morphometric classes resulting in fuzzy areas of peakness associated with summits. Although the result is a raster representation of fuzzy regions, the applied parameters allow individual peaks to be separated from each other. Recently, a number of methods have been developed which both incorporate *a priori* knowledge and are object-based [3, 32]. Their methods use peak contributing areas and prominence to delineate mountains and hills and ranges.

2.2 Determining Region Boundaries

In identifying the borders of any region, or to be more specific in our case, landform, it is important to consider the nature of the region and its borders. Landforms are generally classical examples of *fiat* objects – that is to say they are defined by human perception and do not have a physically unambiguous expression on the earth's surface because they are vague [33]. Thus, regarding our case study, the area which is unambiguously Gürbe valley cannot, by definition, be defined. Recent work has sought to define the boundaries of similar vague fiat regions for so-called vernacular regions, regions which are used in common parlance but have no official or administrative boundary. Examples of such regions include *downtown* or the American Mid-West. Montello et al. [34] investigated this problem by asking residents of Santa Barbara to sketch the boundaries of downtown on a map. More recently, Jones et al. [35] searched for place names co-occurring with vernacular regions, and used density surfaces to estimate the borders of the fiat regions. Both of these sets of techniques used human perception of the boundaries, or locations found inside a region, to delineate a spatial extent for vernacular regions.

2.3 Defining Valleys

There is a range of definitions for the term "valley" in the literature. Here we give three typical examples:

1. a low area more or less enclosed by hills [36];
2. a long, narrow depression in the Earth's surface, usually with a fairly regular downslope (Spatial Data Transfer Standard; [37, 38]); and
3. (a) any low-lying land bordered by higher ground; especially an elongate, relatively large, gently sloping depression of the Earth's surface, commonly situated between two mountains or between ranges of hills or mountains, and often containing a stream with an outlet. It is usually developed by stream erosion, but may be formed by faulting. (b) a broad area of generally flat land extending inland for a considerable distance, drained or watered by a large river and its tributaries; a river basin. Example: the Mississippi Valley [39].

As opposed to the extremely general notion of (1), (2) specifies the shape of the valley explicitly as "long" and "narrow". Additionally, a valley "usually" has a "fairly regular downslope". (3) begins similarly to (2) but then gives some detail, for example, the gentle slope and the presence of streams.
According to the above definitions, characteristics of valleys include the following:

- Valleys are low areas or depressions relative to their surroundings.
- Valleys are elongated.
- Valleys are (gently) sloping.
- Valleys often contain a stream or a river.

The terms 'valley floor' or 'valley bottom' appear infrequently in the literature. However, the Dictionary of Earth Science [40] characterises a valley floor as "the broad, flat bottom of a valley" and says it is "also known as valley bottom; valley plain". Bates and Jackson [39] define it as "the comparatively broad, flat bottom of a valley; (...)" and refer to "valley bottom" and "flood plain" as synonyms. However, 'flood plain' has the implication/affordance of being occasionally inundated by a river (and thus implies that a valley floor must, in contrast to the above, contain a river). In conclusion we can say that a valley floor is a relatively broad, flat region within a valley and will thus inherit the characteristics of valleys listed above.

2.4 Delineating Valleys

Researchers from several fields have investigated methods to extract valleys or features pertaining to valleys from digital representations.

Tribe [41] aimed to automatically recognise valley heads from DEMs by application of a region growing algorithm on seed cells near the upper end of simulated drainage branches. In a follow-up paper, Tribe [42] reviewed shortcomings of existing "valley and drainage network recognition" methods. Most of the reviewed methods seem to yield one pixel wide 'valleys'. A new method improving upon the methodology by Carroll [43] is proposed, including a threshold slope in order to eliminate insignificant depressions and including a larger user-defined neighbourhood in order to reduce network discontinuities in wide, flat-floored valleys.

Miliaresis and Argialas [44] also applied a gradient-dependent region growing algorithm for their delineation of mountains, piedmont slopes and basins from GTOPO30. They used pixels with higher-than-mean flow accumulation as seed cells for basins and, with upslope flow direction, for mountains. However, "the seeds for basins did not give the impression of a network" [44: 720], since basins had gradients less than 2° and aspect/flow direction was undefined. "Thus, the high order valley lines remained undetected" (ibid.). However, the resulting segmentation seems to have overcome this limitation. It was favourably compared to a physiographic map of the region. The extraction of mountain objects but not of basins and slopes was then successfully tested in two additional regions and later re-used in another study [45] which aimed at further describing the extracted mountain-objects with additional topographic attributes (cf. also [46]).

Chorowicz et al. [47] proposed a method for the extraction of drainage networks of areal features. The method seeks to combine a threshold-based "profile scan" and the "hydrological flow routing" method to overcome the problem of hydrological flow routing yielding one-pixel wide channel networks.

Sagar et al. [48] studied the extraction of what they term ridge and valley connectivity networks (RCN and VCN). The authors use multi-scale opening and closing, as well as erosion and dilation of the DEM to extract these networks. While the results for the RCN look relatively sensible, the method seems to have problems with flat-floored valleys where, for smaller neighbourhoods, the concave areas where the valley floor joins the valley sides seem to be extracted rather than the valley axes.

A very different, contour-based approach to hill and valley extraction was proposed by Cronin [49]. One problem of contour-based delineation is the ambiguity of open contours. This is resolved by arbitrarily choosing the smaller area on either side of the open contour as the interior of the contour line. The extraction method is exemplified at four sites. However, three of them feature hills and valleys of approximately half the size of the study area and the fourth example seems to suggest that the presented algorithm tends to derive hills and valleys of a size that is controlled by the map extent and scale.

As already described, curvature-based methods have been implemented by several authors (e.g. [14, 2]) on a multi-scale basis – operationalised as varying window sizes for curvature calculation. However, the latter study focused on mountains or convexities rather than valleys. While these multi-scale methods account for the fuzziness of landforms they generate pixel-based characterisations ('channelness') rather than contiguous objects such as valleys or valley floors. Also, while the multi-scale nature of the approaches is better able to portray landscapes with their inherent multi-scale properties, the problem of choosing an appropriate window size (or range of sizes) for characterisation is unsolved. Gallant and Dowling applied a similar method, but based on the application of slope (representing flatness) and elevation percentiles (representing lowness with respect to surroundings), to the classification of valley bottoms [50].

2.5 Research Gaps

In general, work on the delineation of landforms and landform elements has focused on bottom-up methods, often pixel-based, and especially the delineation of topographic eminences. In this paper, we therefore wish to address the issue of the extraction of valley (floors) from two perspectives – one based around Naïve Geography and the other focusing on a top-down geomorphometric approach. Furthermore, we wish to compare the results of the applied method to a pixel-based method, the classification into six geomorphometric classes identified by Wood [14].

3 Defining the Gürbe Valley through Naïve Geography

For Naïve Geography delineations of the Gürbe valley we looked primarily at natural language descriptions from internet sources provided by both the general public and a tourism organisation in the area. They thus deliberately do not portray a specialist or geoscientific view of the valley or of valleys in general.

The general public's view was operationalised using Wikipedia. Although the community model ('crowd-sourcing') of this online reference work has limitations, Wikipedia is used and referred to by the public. Wikipedia is split into language groups, the encyclopaedic coverage and, of course, regional focus of the language groups differing significantly. There were 2,150,000 English articles vs. 690,000 German articles as of January 7th, 2008 [51].

For the tourism perspective we referred to the tourism association of the Gürbe valley (Verkehrsverband Region Gürbetal, [52]) which owns the internet domain 'www.guerbetal.ch'. We obtained a snapshot of the website as of February 2nd, 2007 from the internet archive [53].

In order to gain additional clues on the extent of the Gürbe valley and some other topographic features mentioned e.g. by Wikipedia, we analysed toponyms used in Swiss topographic maps, similarly to [2]. For this purpose we used three scales of Swiss maps: 1:25,000, 1:50,000 and 1:100,000.

For comparison with DEM-based methods, and due to the limited number of points, convex hulls were derived for toponym label locations associated with the Gürbe valley, whilst region boundaries were used *as is*.

4 Automatically Extracting Valley Floors

4.1 Operationalisation

In developing a method for the extraction of valleys, the eventual aim of our work, Maxwell [1] was chosen as a starting point. The dales as defined by Maxwell equal drainage basins and effectively enclose valleys. While the enclosing relation may be one-to-one (typically in small headwater drainage basins), this is of course not necessarily the case for larger drainage basins which may contain several valleys. Thus, in order to narrow down the search area for valleys, which we consider to have a one-to-one relationship with valley floors, we clip drainage basins of a certain Shreve order with contributing drainage basins of lower orders (cf. also [54]). A drainage sub-basin is thus defined, a core area more closely related to one valley than the original drainage basin.

Starting from the definitions in section 2.3 we assumed that streams or thalwegs could well serve as conceptual cores of valleys and their floors. Valley floors can then be described as relatively flat areas bordering thalwegs. Thus, valley floors can be extracted by imposing a gradient threshold on a region growing procedure seeded by thalweg/stream cells. In accordance with our assertions on the relations of drainage (sub-)basins and valleys we also imposed drainage sub-basin constraints – region growing only occurs within, and not across, sub-basins.

4.2 Implementation

The procedure of extracting approximations to valley floors is as follows. First, the SRTM DEM [55] was projected into the Swiss national projection system and resampled to 100 m resolution. The DEM was then filled and D8 flow directions and a flow accumulation grid calculated. By imposing a channel initiation threshold of ≥ 500 cells a stream network and its Shreve ordering was derived, with pourpoints being created where stream of differing orders merged.

Subsequently, drainage basins of order x were clipped by all drainage basins of order $y < x$. The use of Shreve ordering led to a segmentation of the drainage basins in general flow direction, i.e. each segment of a river between two tributaries has its own drainage sub-basin, cf. Fig. 1. A raster dataset was computed storing for each drainage sub-basin its hydrological order and an ID unique amongst the sub-basins of that order.

Using this raster and a raster of the streams a region growing procedure using stream cells as seeds was carried out. Growing was allowed to occur only within an

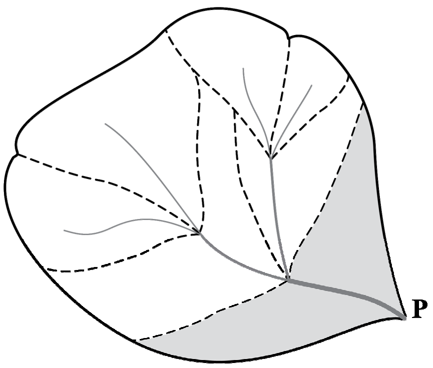


Fig. 1. Clipping of drainage basins. Solid outline represents original drainage basin of point P, dashed outlines represent several drainage sub-basins pertaining to different streams (grey lines). The drainage sub-basin of point P is represented by the grey area.

individual drainage sub-basin. A raster cell *i* was classified as pertaining to the valley floor, when at least one of its neighbours was a seed cell or a grown valley floor cell and one of the following conditions concerning the elevations of cell *i* and the seed cell was met:

$$\text{Cardinal neighbours: } \tan(\gamma_{thrsh}) \cdot \lambda \geq elev_i - elev_{seed} \geq 0m\,. \tag{1}$$

$$\text{Diagonal neighbours: } \tan(\gamma_{thrsh}) \cdot \sqrt{2} \cdot \lambda \geq elev_i - elev_{seed} \geq 0m\,. \tag{2}$$

where γ_{thrsh}: gradient threshold [°], λ: cell size [m], $elev_i$ and $elev_{seed}$: elevation [m] of cell *i* and seed cell, respectively.

This procedure ensures that valley floors are contiguous and that only those areas which can be reached from the thalweg with a low slope are delineated as belonging to the valley floor, thus matching the definitions for valley floors in section 2.3. Region growing was run iteratively until no new valley floor cells were detected. We tested a range of gradient thresholds (γ_{thrsh}) from 0.25° to 3° where, through qualitative visual examination, a threshold value of 1.5° gave the most promising results and was used in the following evaluation.

5 Results and Discussion

Fig. 2 shows delineated valley floors in Switzerland and bordering regions. Note the floors of the broader alpine valleys, the conspicuous Rhine valley near the border of

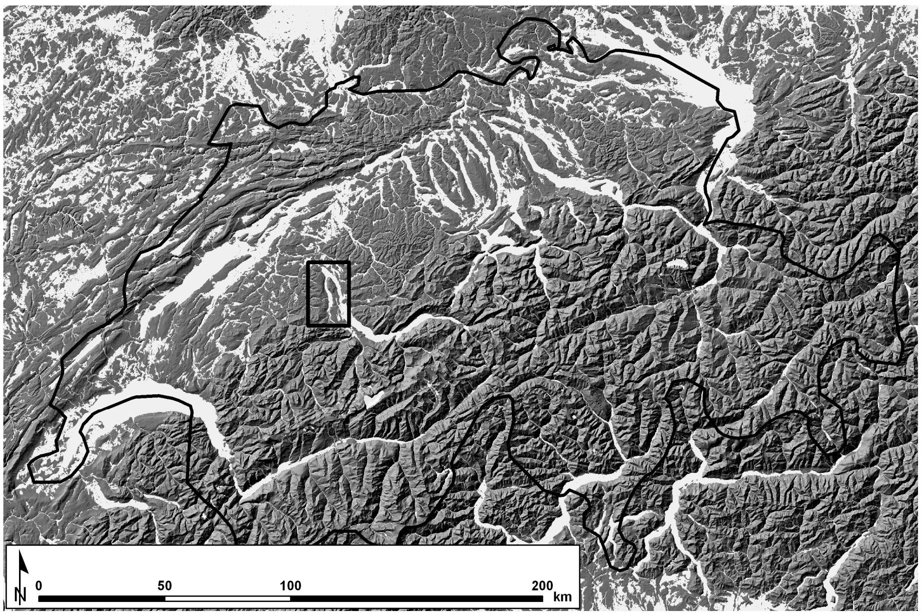


Fig. 2. Delineation of valley floors (light grey areas) using 1.5° threshold in the area of Switzerland (border in black). The black square denotes the region of the Gürbe (and Aar) valley subsequently analysed in detail.

Switzerland, Liechtenstein and Austria in the upper right corner and the Rhône valley in south-western Switzerland. Note also the floor of the Rhine Graben marking the border of France and Germany. While the extents of valley floors in the Swiss Prealps and in the lowland seem relatively sensible, the delineation may be problematic in France near the western border of the study area. There an obviously less accentuated topography leads to large regions being classified as valley floor.

In the remainder of this section we will compare the extent of the delineated valley floor in the Gürbe valley – a prealpine valley marked by the square in Fig. 2 – to valley delineations based on Naïve Geography descriptions of the area. Subsequently we compare our valley floor delineation to the distribution of six morphometric classes [14].

5.1 Comparison with Naïve Geography Delineations

The following excerpt is our translation of the entry in the German-speaking Wikipedia article "Gürbetal" (Gürbe valley) [56]:

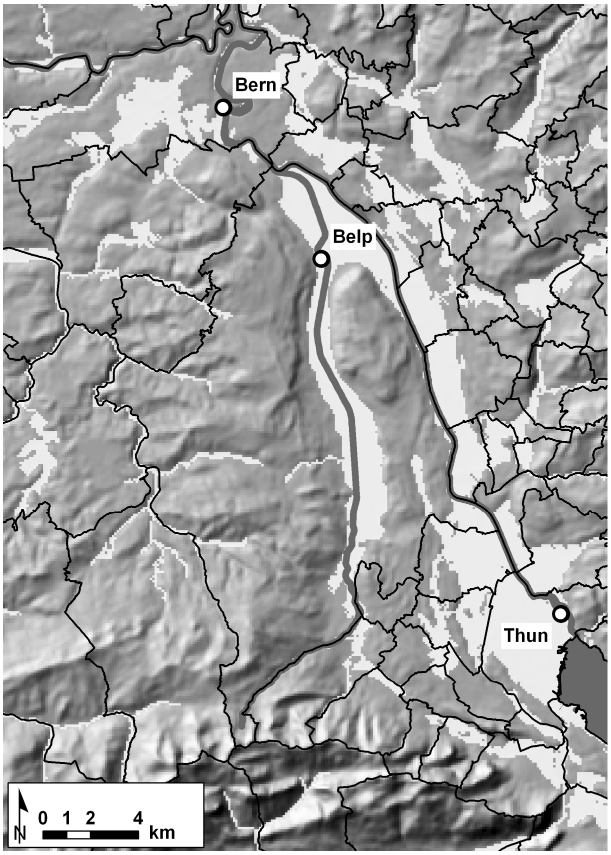


Fig. 3. Characterisation of the Gürbe valley in the German-speaking Wikipedia. Black linear features are administrative boundaries (large, in the middle: district of Seftigen; smaller: adjacent municipalities), dark grey features are water bodies. Background is a hillshaded DEM with delineated valley floors superimposed in light grey.

"The Gürbe valley is the region between Bern and Thun (west of the Aar) in Switzerland. It encompasses the district of Seftigen and neighbouring municipalities. The valley is named after the river Gürbe. The largest town in the valley is Belp. The Gürbe and Aar valleys are separated by Belpberg (a ridge). To the west, the Gürbe valley is bordered by Längenberg. The flat Gürbe valley floor has a width of between 1 and 2 km and is intensively farmed."

Fig. 3 shows a depiction of the most important elements in the Wikipedia article along with our delineation of the valley floor. In the western part of the area is the River Gürbe, in the eastern part the river Aar flows out of Lake Thun. North of Belp the Gürbe flows into the Aar which then in turn flows through Bern. As can be seen in Fig. 3, Wikipedia's description of the ridge Belpberg separating the Gürbe valley from the Aar valley somewhat contradicts the assertion that the Gürbe valley is the region bordering the Aar from the west or encompasses the district of Seftigen (whose eastern border is in fact the Aar). However, the width of the Gürbe valley specified by Wikipedia to be 1 to 2 km closely matches the area the DEM-based algorithm delineated as valley floor.

The boxes in Fig. 4 denote the extent of toponym labels mentioned in the Wikipedia article [56] signifying the Längenberg, the Gürbe valley and the Belpberg (from east to west) as extracted from Swiss 1:25,000, 1:50,000 and 1:100,000 maps. Note how the Belpberg toponym labels indeed flank those of the Gürbe valley and the adjacent delineated valley floor of the Aar valley. The boundary of the district of Seftigen, however, contains Belpberg and can thus be deemed to be – at least in this region – too wide an approximation to the Gürbe valley.

Although for cartographic reasons toponym labels may not be placed directly over the objects they signify, toponym label locations and the valley floor delineated using our algorithm coincide well. However, the 1:100,000 toponym label of Gürbe valley extends significantly further south than toponym labels from larger scales into a region our algorithm also delineated as valley floor.

The apparent uncertainty about the upper end of the Gürbe valley is reinforced by descriptions by the tourism authority of the Gürbe valley. Its website [52, 53] lists seventeen municipalities that belong to the Gürbe valley which are shown in Fig. 5 along with their convex hull. This delineation contains large areas of the delineated valley floor and also matches relatively closely the locations of the Gürbe valley toponyms in Fig. 4 – except for the toponym of 1:100,000 which extends considerably further south and the municipality of Rüeggisberg which, judged from the toponyms (Fig. 4) is west of Längenberg.

5.2 Comparison with Pixel-Based Morphometric Classification

In order to compare the valley floor delineation method with a pixel-based classification, we computed six morphometric classes [14] for the whole region shown in Fig. 2. We selected a window for implicit surface fitting of between 3 and 7 cells (~300 to 700 metres). In order to determine sensible thresholds for surface gradient and curvature we computed classifications using LANDSERF [57], with curvature threshold of 0.1 and 0.5. We selected a gradient threshold of 1.5° which both gave sensible results and matches the threshold of our region-growing approach.

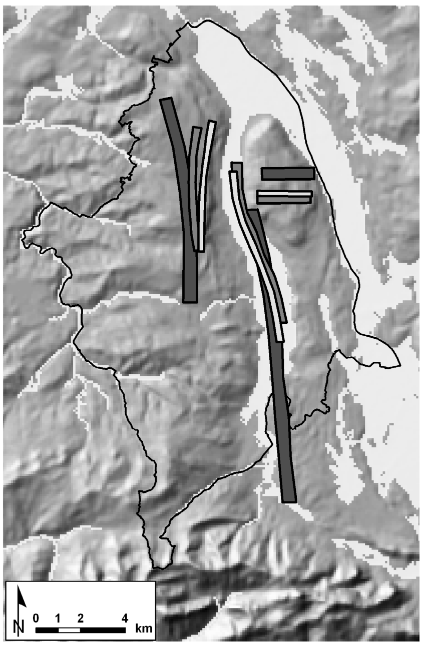


Fig. 4. Outline of toponym labels of Swisstopo maps 1:25,000 (lightest), 1:50,000 (medium) and 1:100,000 (darkest grey) referring to Längenberg (west), Gürbe valley (middle) and Belpberg (east). Background: hillshaded DEM, and delineated valley floor, district of Seftigen (black outline) for reference.

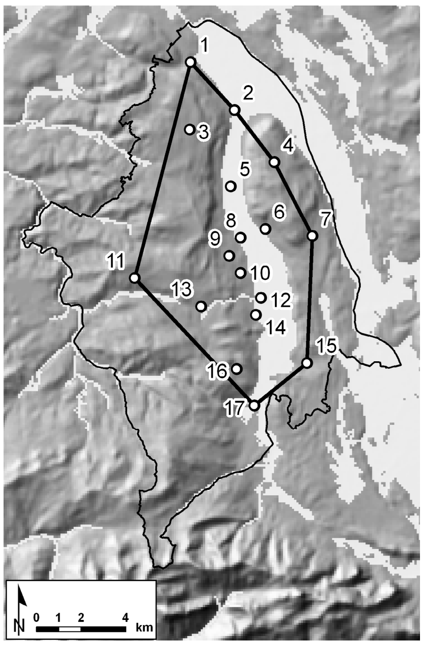


Fig. 5. Municipalities listed as belonging to the Gürbe valley by the tourism organisation of the region together with a convex hull (1: Kehrsatz, 2: Belp, 3: Zimmerwald, 4: Belpberg, 5: Toffen, 6: Gelterfingen, 7: Gerzensee, 8: Kaufdorf, 9: Rümligen, 10: Kirchenthurnen, 11: Rüeggisberg, 12: Mühleturnen, 13: Riggisberg, 14: Lohnstorf, 15: Seftigen, 16: Burgistein, 17: Wattenwil). Background as in Fig. 4

Table 1. Proportions of morphometric classes for region shown in Fig. 2

	Thresholds in classification	
	{1.5°; 0.1}	{1.5°; 0.5}
Pit	0.28 %	0.00 %
Channel	28.93 %	6.64 %
Pass	1.00 %	0.01 %
Ridge	28.27 %	8.03 %
Peak	0.22 %	0.00 %
Planar	41.30 %	85.31 %

Table 1 shows the proportions of each morphometric class for the two curvature values. With a lower threshold curvature the proportion of curved features such as channels, passes or ridges is considerably higher. The adoption of a higher threshold curvature results in an explosion in planar features (85% of the whole region is classified as planar) and we do not further compare values with these thresholds.

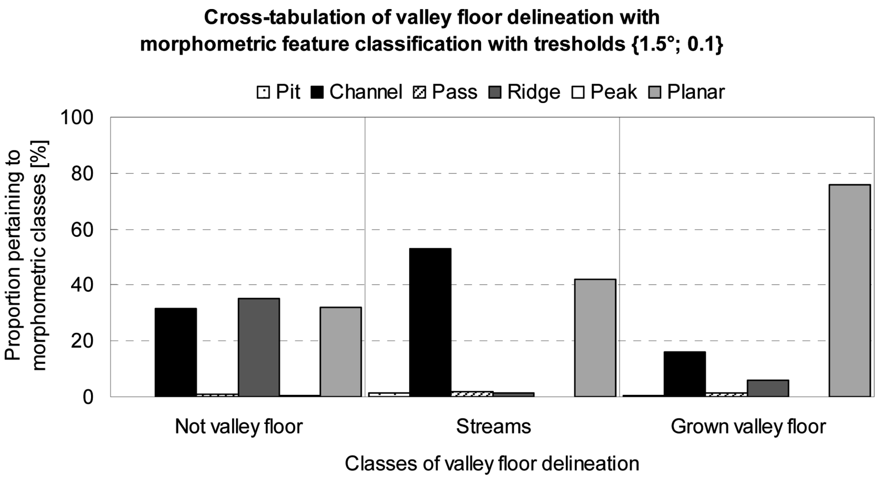


Fig. 6. Cross-tabulation statistics of valley floor delineation vs. morphometric feature classification for region shown in Fig. 2 with gradient threshold of 1.5° and curvature threshold 0.1 in the latter

Fig. 6 shows a cross-tabulation between our classification into streams, valley floor and areas not deemed to be valley floor and the six-fold morphometric classification into pit, channel, pass, ridge, peak and planar classes with thresholds {1.5°; 0.1}. Areas not classified as valley floors have almost equal proportions of channel, ridge and planar pixels. This is in clear contrast to streams where channel pixels dominate and almost no ridge pixels (< 2%) occur and to the valley floors where there is a clear dominance of planar pixels, followed by channel (16%) and ridge (6%) pixels.

Figs. 7 and 8 show the spatial arrangement of the delineated valley floor with respect to the morphometric classification and are mutually exclusive since they each depict a part of the six-fold classification but together show all classes except for passes (which are also present in the valley floor).

Fig. 7 shows many channel features on the valley floor, however, their location suggests that these are primarily artifacts occurring near the concavity of the transition from valley floor to side slopes. Pits are found throughout the valley floor, often close to channel features. Fig. 8 shows that there are several instances of ridge pixels and also some peak pixels located within the delineated valley floor mainly (but not exclusively) of the Aar valley. These stem from minor surface undulations which were, from the perspective of some seed pixels, sufficiently smooth to be classified as valley floor.

Summarising, the classifications of morphometric features suggest that the attributes of our valley floors at a pixel level make sense (relative dominance of channel and planar features in streams and valley floors). Furthermore, minor ridges and peaks (which may well be glacial features such as moraines and eskers) are identified by our algorithm as belonging to the valley floor. This suggests a potential strength of object-based top-down approaches over pixel-based methods, where the delineation of a relatively simple landform such as valley floor may not easily be reproduced by extending a pixel-based morphometric classification (e.g. through subsequent application of a gradient threshold on planar features).

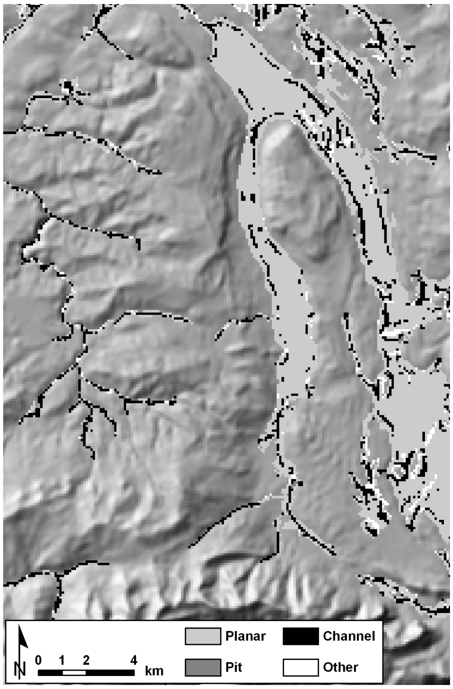


Fig. 7. Planar, pit and channel pixels of classification {1.5°; 0.1} within delineated valley floors

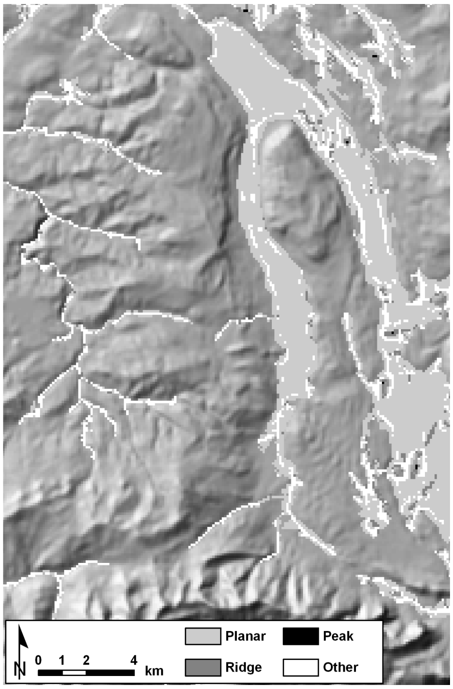


Fig. 8. Planar, ridge and peak pixels of classification {1.5°; 0.1} within delineated valley floors

5.3 Limitations and Extensibility of the Approach

An obvious limitation of the approach is the adoption of a single universal gradient threshold for the delineation of valley floors with a region growing algorithm. While the quality of the results can be judged visually, there is no clear indication of a universally applicable threshold to be obtained from the literature or everywhere else. A possible extension of the approach would select a threshold based upon some contextual information, e.g. lower gradient threshold for lower order (and usually less incised) streams or the tuning of the threshold with some property of the respective drainage sub-basin. However, while such a procedure might improve results it would also introduce additional ambiguity in the form of new parameters.

6 Conclusions and Outlook

In this paper our key aim was to develop a robust method, capable of deriving valley floor extents over a large area. The developed method is object-based and top-down – that is to say it uses definitions of valley floors in the algorithm development and grows regions which are considered to be valley floor. To assess the method, given the *fiat* nature of landforms, we compared the extents of valleys derived from Naïve

Geography sources with valley floors from our algorithm. Using the Gürbe valley in Switzerland as an example, comparisons show a relatively good agreement between the vernacular region associated with the Gürbe valley from a variety of sources and the valley floor delineated using our DEM-based algorithm. Additionally, we compared our top-down approach to a pixel-based more bottom-up approach which classifies a DEM into six morphometric classes. This comparison showed that our valley floors had differing distributions of morphometric classes from non-valley floor areas (primarily planar slopes and channels), though our algorithm was capable of classifying pixels identified as ridges and peaks as belonging to the valley floor.

It appears that valleys and associated landforms or, more generally, topographic depressions, have gained less attention in the literature than, for example, topographic eminences. We thus intend to continue our current work to delineate valley sides, and thus define the extent of valleys and their relationship to topographic eminences.

Acknowledgements. The research reported in this paper was partially funded by the project TRIPOD supported by the European Commission under contract 045335.

We thank the canton of Bern [58] for geographic datasets displayed in Figs. 3 to 5. Administrative boundaries (districts and municipalities) in Figs. 3 to 5: “Political boundaries of the canton of Bern 1:5,000”, © Geoinformation Office of the canton of Bern. Lakes and rivers in Fig. 3: “Digital map of water body protection of the canton of Bern (GSK25)”, © Hydrology Office of the canton of Bern and “Hydrologic network of the canton of Bern 1:5,000”, © Civil Engineering Office of the canton of Bern, respectively.

References

1. Maxwell, J.C.: On Hills and Dales. Philosophical Magazine (1870); reprinted In: Niven, W.D. (ed.) The Scientific Papers of James Clerk Maxwell, pp. 233–240. Dover Publications, New York (1965)
2. Fisher, P., Wood, J., Cheng, T.: Where Is Helvellyn? Fuzziness of Multi-scale Landscape Morphometry. Transactions of the Institute of British Geographers 29, 106–128 (2004)
3. Chaudhry, O.Z., Mackaness, W.A.: Creating Mountains out of Mole Hills: Automatic Identification of Hills and Ranges Using Morphometric Analysis. Transactions in GIS (in press)
4. Mark, D., Sinha, G.: Ontology of landforms: Delimitation and Classification of Topographic Eminences. In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 129–132. Springer, Heidelberg (2006)
5. Hugget, R.J.: Fundamentals of Geomorphology. Routledge, London (2007)
6. Whittow, J.B.: “Landform.” The Penguin Dictionary of Physical Geography. Penguin Books, London (2000)
7. Lapidus, D.F., Coates, D.R., Winstanley, I., MacDonald, J., Burton, C.: “Landform.” Collins Dictionary of Geology. HarperCollins Publishers, Glasgow (2003)
8. Blasczcinsky, J.S.: Landform Characterization with Geographic Information Systems. Photogrammetric Engineering and Remote Sensing 63, 183–191 (1997)
9. Evans, I.S.: General Geomorphometry, Derivatives of Altitude, and Descriptive Statistics. In: Chorley, R.J. (ed.) Spatial Analysis in Geomorphology, pp. 17–90. Methuen & Co. Ltd., London (1972)

10. Moore, I.D., Grayson, R.B., Ladson, A.R.: Digital Terrain Modelling: A Review of Hydrological, Geomorphological, and Biological Applications. Hydrological Processes 5, 3–30 (1991)
11. Zevenbergen, L.W., Thorne, C.R.: Quantitative Analysis of Land Surface Topography. Earth Surface Processes and Landforms 12, 47–56 (1987)
12. Kienzle, S.W.: The Effect of DEM Raster Resolution on First Order, Second Order and Compound Terrain Derivatives. Transactions in GIS 8, 83–111 (2004)
13. Shary, P.A., Sharaya, L.S., Mitusov, A.V.: Fundamental Quantitative Methods of Land Surface Analysis. Geoderma 107, 1–32 (2002)
14. Wood, J.: The Geomorphological Characterisation of Digital Elevation Models. PhD thesis, University of Leicester, UK (1996)
15. Mark, D.M.: Geomorphometric Parameters: A Review and Evaluation. Geografiska Annaler, Series A, Physical Geography 57, 165–177 (1975)
16. Thompson, J.A., Bell, J.C., Butler, C.A.: Digital Elevation Model Resolution: Effects on Terrain Attribute Calculation and Quantitative Soil-Landscape Modeling. Geoderma 100, 67–89 (2001)
17. Burrough, P.A., van Gaans, P.F.M., MacMillan, R.A.: High-Resolution Landform Classification Using Fuzzy k-Means. Fuzzy Sets and Systems 113, 37–52 (2000)
18. Beven, K.J., Kirkby, M.J.: A Physically Based, Variable Contributing Area Model of Basin Hydrology. Hydrological Sciences Bulletin 24, 43–69 (1979)
19. Quinn, P.F., Beven, K.J., Lamb, R.: The ln(a/tan β) Index: How to Calculate It and How to Use It within the TOPMODEL Framework. Hydrological Processes 9, 161–182 (1995)
20. de Bruin, S., Stein, A.: Soil-Landscape Modelling Using Fuzzy c-Means Clustering of Attribute Data Derived from a Digital Elevation Model (DEM). Geoderma 83, 17–33 (1998)
21. Burrough, P.A., Wilson, J.P., van Gaans, P.F., Hansen, A.J.: Fuzzy k-Means Classification of Topo-climatic Data as an Aid to Forest Mapping in the Greater Yellowstone Area, USA. Landscape Ecology 16, 523–546 (2001)
22. Pennock, D.J., Zebarth, B.J., De Jong, E.: Landform Classification and Soil Distribution in Hummocky Terrain, Saskatchewan, Canada. Geoderma 40, 297–315 (1987)
23. Ruhe, R.V.: Elements of the Soil Landscape. In: Transactions of the 7th International Congress of Soil Science, International Society of Soil Science, pp. 165–170 (1960)
24. Dikau, R.: The Application of Digital Relief Model to Landform Analysis in Geomorphology. In: Raper, J. (ed.) Three Dimensional Applications in Geographical Information Systems, pp. 51–77. Taylor & Francis, London (1989)
25. Pennock, D.J., Corre, M.D.: Development and Application of Landform Segmentation Procedures. Soil & Tillage Research 58, 151–162 (2001)
26. Pennock, D.J.: Terrain Attributes, Landform Segmentation, and Soil Redistribution. Soil & Tillage Research 69, 15–26 (2003)
27. Moreno, M., Levachkine, S., Torres, M., Quintero, R.: Landform Classification in Raster Geo-images. In: Sanfeliu, A., Trinidad, J.F.M., Ochoa, J.A.C. (eds.) CIARP 2004. LNCS, vol. 3287, pp. 558–565. Springer, Heidelberg (2004)
28. Bolongaro-Crevenna, A., Torres-Rodríguez, V., Sorani, V., Frame, D., Ortiz, M.A.: Geomorphometric Analysis for Characterizing Landforms in Morelos State, Mexico. Geomorphology 67, 407–422 (2005)
29. MacMillan, R.A., Pettapiece, W.W., Nolan, S., Goddard, T.W.: A Generic Procedure for Automatically Segmenting Landforms into Landform Elements Using DEMs, Heuristic Rules and Fuzzy Logic. Fuzzy Sets and Systems 113, 81–109 (2000)
30. Schmidt, J., Hewitt, A.: Fuzzy Land Element Classification from DTMs Based on Geometry and Terrain Position. Geoderma 121, 243–256 (2004)

31. Lucieer, A., Stein, A.: Texture-Based Landform Segmentation of LiDAR Imagery. International Journal of Applied Earth Observation and Geoinformation 6, 261–270 (2005)
32. Greatbach, I., Wood, J., Fisher, P.: A Comparison of Morphometric and Web Prominence of Mountain Features. In: Geographical Information Science Research UK (GISRUK) Conference, NUI Maynooth, Ireland, pp. 312–317 (2007)
33. Smith, B., Varzi, A.C.: Fiat and Bona Fide Boundaries. Philosophy and Phenomenological Research 60, 401–420 (2000)
34. Montello, D.R., Goodchild, M.F., Gottsegen, J., Fohl, P.: Where's Downtown? Behavioral Methods for Determining Referents of Vague Spatial Queries. Spatial Cognition & Computation 3, 185–204 (2003)
35. Jones, C.B., Purves, R.S., Clough, P.D., Joho, H.: Modelling Vague Places with Knowledge from the Web. International Journal of Geographic Information Science (in press)
36. Ordnance Survey Ontologies,
`http://www.ordnancesurvey.co.uk/oswebsite/ontology`
37. USGS: SDTS – View the Standard,
`http://mcmcweb.er.usgs.gov/sdts/standard.html`
38. ANSI NCITS 320-1998. Draft. Part 2: Spatial Features,
`http://thor-f5.er.usgs.gov/`
`sdts/standard/latest_draft/pdf/part2.pdf`
39. Bates, R.L., Jackson, J.A. (eds.): "Valley."; "Valley floor." Glossary of Geology. American Geological Institute, Alexandria (1990)
40. McGraw-Hill Dictionary of Earth Science. "Valley floor." McGraw-Hill, New York (2003)
41. Tribe, A.: Automated recognition of valley heads from digital elevation models. Earth Surface Processes and Landforms 16, 33–49 (1991)
42. Tribe, A.: Problems in automated recognition of valley features from digital elevation models and a new method toward their resolution. Earth Surface Processes and Landforms 17, 437–454 (1992)
43. Carroll, R.: Automated gully delineation using digital elevation data. ACSM-ASP 49th Annual Meeting, Technical Papers, Washington D.C., USA, pp. 144–151 (1983)
44. Miliaresis, G.C., Argialas, D.P.: Segmentation of Physiographic Features from the Global Digital Elevation Model/GTOPO30. Computers & Geosciences 25, 715–728 (1999)
45. Miliaresis, G.C., Argialas, D.P.: Quantitative Representation of Mountain Objects Extracted from the Global Digital Elevation Model (GTOPO30). International Journal of Remote Sensing 23, 949–964 (2002)
46. Miliaresis, G.C.: Geomorphometric Mapping of Asia Minor from GLOBE Digital Elevation Model. Geografiska Annaler, Series A, Physical Geography 88, 209–221 (2006)
47. Chorowicz, J., Ichoku, C., Riazanoff, S., Kim, Y., Cervelle, B.: A combined algorithm for automated drainage network extraction. Water Resources Research 28, 1293–1302 (1992)
48. Sagar, B.S.D., Murthy, M.B.R., Rao, C.B., Raj, B.: Morphological approach to extract ridge and valley connectivity networks from Digital Elevation Models. International Journal of Remote Sensing 24, 573–581 (2003)
49. Cronin, T.: Classifying hills and valleys in digitized terrain. Photogrammetric Engineering and Remote Sensing 66, 1129–1137 (2000)
50. Gallant, J.C., Dowling, T.I.: A multiresolution index of valley bottom flatness for mapping depositional areas. Water Resources Research 39, 1347 (2003)
51. Wikipedia, `http://www.wikipedia.org`
52. Verkehrsverband Region Gürbetal, `http://www.guerbetal.ch`
53. Internet Archive, `http://www.archive.org`

54. Demoulin, A., Bovy, B., Rixhon, G., Cornet, Y.: An automated method to extract fluvial terraces from digital elevation models: The Vesdre valley, a case study in eastern Belgium. Geomorphology 91, 51–64 (2007)
55. Jarvis, A., Reuter, H.I., Nelson, A., Guevara, E.: Hole-filled seamless SRTM data V3, International Centre for Tropical Agriculture (CIAT), `http://srtm.csi.cgiar.org`
56. Wikipedia, D.E.: "Gürbetal", `http://de.wikipedia.org/wiki/G%C3%Bcrbetal`
57. LANDSERF, `http://www.landserf.org`
58. Geoinformation Office of the Canton of Bern, `http://www.bve.be.ch/site/index/agi`

Single-Holed Regions: Their Relations and Inferences

Maria Vasardani and Max J. Egenhofer

National Center for Geographic Information and Analysis
and
Department of Spatial Information Science and Engineering
University of Maine
Boardman Hall, Orono, ME 04469-5711, USA
{mvasardani,max}@spatial.maine.edu

Abstract. The discontinuities in boundaries and exteriors that regions with holes expose offer opportunities for inferences that are impossible for regions without holes. A systematic study of the binary relations between single-holed regions shows not only an increase in the number of feasible relations (from eight between two regions without holes to 152 for two single-holed regions), but also identifies the increased reasoning power enabled by the holes. A set of quantitative measures is introduced to compare various composition tables over regions with and without holes. These measures reveal that inferences over relations for holed regions are overall crisper and yield more unique results than relations over regions without holes. Likewise, compositions that involve more holed regions than regions without holes provide crisper inferences, which supports the need for relation models that capture holes explicitly.

1 Introduction

The proliferation of geosensor networks (Stefanidis and Nittel 2004) for monitoring spatially distributed phenomena challenges spatial querying and spatial reasoning methods beyond the state of the art of current GISs. In addition to the real-time analysis of spatial fields, the traditional spatial-database domain of object-based querying requires renewed attention as well. While the focus on simple cases of homogeneous 2-dimensional regions has dominated the formalization of spatial relations and spatial reasoning, *holes*—typically considered the minnows of geospatial modeling—raise to primetime with the analysis of objects that are derived from geosensor networks. In such settings, the need for spatial reasoning with holes is not solely motivated by the prototypical static cases such as South Africa completely surrounding Lesotho or Manitoulin Island in Lake Huron as the world's largest island in a lake. It also arises from less controlled and more frequently occurring scenarios such as sensor failures, leading to *coverage holes*, that is, areas that cannot be reached by a certain amount of sensors (Ahmed *et al.* 2005). Likewise variations in analytical thresholds to extract regions of homogenous values from fields may yield regions with holes, while other types of holes found in wireless sensor networks, such as routing holes, jamming holes, or sinkholes, are typically related to cavities in graphs. The holes referred to in this paper, more closely resemble coverage holes.

Figure 1 shows a sensor network of buoys deployed in the Gulf of Maine to record such variables as wind speed, wave height, air and water temperatures, and atmospheric pressure (http://www.gomoos.com/). The two snapshots in Figures 1a and 1b refer to the available data on two consecutive days, where the highlighted regions depict two zones with data unavailability, each around a malfunctioning buoy, yielding holes in the coverage regions. With the movement of the buoys on the ocean surface, these zones of data availability shift, and so do the holes of missing data. The comparison of both regions over the 2-day period reveals that for the first buoy, both the region and its hole have moved on the second day (Figure 1c), while the second buoy's region has changed only slightly in size but its hole has moved (Figure 1d). In order to perform such analyses at a more abstract level, so that it could be performed conveniently through a spatial query language, one needs an account of the possible relations between such regions with holes. Beyond such an enumeration of all possible cases, however, the inferences that can be drawn from such relations reveal new insights. For instance, what could be derived if one had further information available that on day 3 the regions have moved such that they are now disjoint from their positions on day 2?

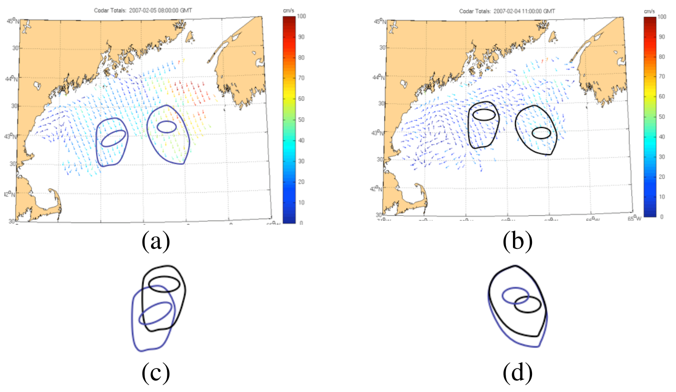

Codar Totals: 2007-02-05 08:00:00 GMT
Codar Totals: 2007-02-04 11:00:00 GMT
cm/s
(a)
(b)
(c)
(d)

Fig. 1. Example of regions with a hole due to data holes in a buoy network in the Gulf of Maine: (a) two regions, A (left) and B (right), with missing data on 02/04/2005, (b) two regions, A' (left) and B' (right), with missing data on 02/05/2005, (c) overlay regions A and A', and (d) overlay of regions B and B'

The nature of holes in three-dimensional objects has been studied from a philosophical perspective (Casati and Varzi 1994; Lewis and Lewis 1970). An explicit theory for holes (Varzi 1996) builds on contributions from ontology (addressing the holes' identity conditions), mereology (their part-whole relations), topology (the relations between holes and their hosts, as well as the patterns of their interactions with the environment), and morphology (referring to the shapes of holes, their fillability, and their ability to be penetrated by other entities). Despite such philosophical studies computational models of spatial relations between regions with holes have been scarce, however.

The most fine-grained models for topological relations involving regions with holes employ a spatial scene of regions (Egenhofer *et al.* 1994). The topological relation between two regions, one with *n* holes the other with *m* holes, is modeled as a spatial scene with 2+*n*+*m* regions, in which each holed region is filled, yielding a generalized region. All holes are *inside* their hosts, the generalized region (although variations with holes *coveredBy* or *equal* to the host have been considered as well), and all other relations with respect to the generalized regions and between the *n*+*m* holes are captured explicitly. Based on this approach, the topological relations between a region with a hole and a region without a hole have been studied (Egenhofer and Vasardani 2007).

This paper investigates how spatial-relation reasoning over 2-dimensional regions with holes differs from similar inferences without holes. The introduction of a hole into a region is expected to increase the number of possible relations with such a holed-region. For instance, in $\mathbb{R}^2$, a hole in a region gives rise to two cases in which a region is separate from another region (Figures 2a and 2b) and three qualitatively different cases in which a region touches another region (Figures 2c-2e). A question of interest is to examine how the larger number of relations affects the ambiguity of qualitative inferences through the relations' compositions.

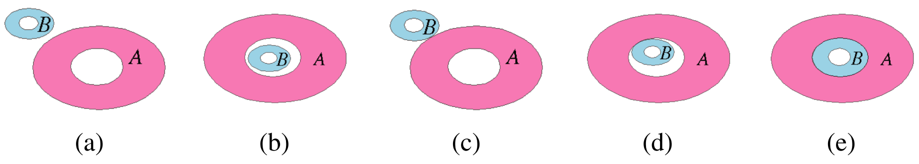


Fig. 2. Example configurations of binary relations of single-holed regions

For this goal, this paper derives the set of topological relations between two regions, each with one hole, called a *single-holed region*, and the complete composition table for relations between single-holed region relations. The crispness and ambiguity of these inferences is then compared with the established inferences for the relations of regions without holes, called simple regions (Egenhofer 1994) and for the relations between a simple region and a single-holed region (Egenhofer and Vasardani 2007). This analysis calls for new analytical methods for comparing composition tables over relation sets with different cardinalities, which are developed in this paper.

The remainder of the paper is structured as follows: Section 2 discusses various models of relations between regions with holes and visually similar configurations. A set of jointly exhaustive and mutually disjoint binary topological relations between two single-holed regions is systematically derived (Section 3) and analyzed for their properties (Section 4). Section 5 determines the inferences that one obtains from the composition of two single-holed region relations. Section 6 introduces a novel set of quantitative measures to compare composition tables over differing domains and applies them to the various compositions over single-holed regions and regions without holes. The paper closes with conclusions and a discussion of future work (Section 7).

2 Relations of Holed Objects

Models of spatial relations account for holes at various granularities. Some address relations with holed objects at the same level as without holes, essentially ignoring the restrictions, as well as opportunities, offered by the holes when making qualitative inferences. Other models account for holes, but employ the same mechanisms developed successfully for relations of objects without holes. Most restrictive are models, such as the 9-intersection (Egenhofer and Herring 1994), that exclude regions with holes, but in return its region relations form a relation algebra with consistent inference behavior, a property that is unknown for other sets of spatial relations with a more ambitious domain (Li and Ying, 2003).

The coarsest level of topological relations that apply to holed regions abstracts away the holes, considering only relations between regions without holes; therefore, no particularities that arise through the holes can be captured (such as a region that is fully contained in another regions' hole vs. in the region). This applies, for instance, to the relations between minimum bounding rectangles (Papadias *et al.* 1995).

In a similar way the region-connection-calculus (RCC) considers for regions with holes the same relation definitions as for regions without holes, yielding eight jointly exhaustive and pairwise disjoint topological relations (Randell *et al.* 1992). While compact and compatible across regions with an arbitrary number of holes, this approach implies, for instance, that a region *A* located outside of another region *B* (Figure 2a) has the same topological relation as if *A* were located in *B*'s hole (Figure 2b). In the same vein, the three external-connection configurations—*A* is externally connected to *B* from the outside (Figure 2c) or from the inside through *B*'s hole (Figure 2d), or by filling *B*'s hole completely (Figure 2e)—are categorized by the same RCC relation.

The application of the vanilla 9-intersection to such complex spatial objects as regions with holes yields thirty-three relations (Schneider and Behr 2006). Much like RCC it provides a single relation for configurations with or without holes distinguishing, however, some additional details that are a grouped together in RCC. This comes, however, at the premium of causing some anomalies when holes are introduced into or removed from regions, leading at times to reclassifications of relations. For instance, *B* filling *A*'s hole (Figure 2e) has the same vanilla 9-intersection as Figures 2c and 2d, but when *B*'s hole is filled, the configuration switches to a relation of its own. RCC, on the other hand, offers a more consistent treatment of such scenarios.

Some of the most detailed relations that seem to model holed region relations actually apply to visually similar configurations. These are the models for the relations between broad-boundary regions (Clementini and di Felice 1996) and the egg-yolk model (Cohn and Gott 1996). Broad-boundary regions and egg-yolk regions yield conigurations that resemble those of holed regions if one depicts them graphically by only drawing the regions' outlines. Since they both model imprecise regions, however, they are conceptually significantly distinct from holed regions (Egenhofer and Vasardani 2007). For instance, the minor variations in pre-conditions used for the broad-boundary vs. egg-yolk models, leading to relation sets with different cardinalities, do not apply to holed regions.

3 Topological Relations between Two Single-Holed Regions

The following notation is used for topological relations: t_{RR} refers to the topological relation between two simple regions (i.e., without a hole), whereas $t_{\mathrm{R_hR_h}}$ denotes the topological relation between two single-holed regions. At times the relation between a simple region and a single-holed region, and its converse, denoted by $t_{\mathrm{RR_h}}$ and $t_{\mathrm{R_hR}}$, respectively, will be used as well.

The topological relation between two single-holed regions, A and B, is modeled as a *spatial scene* (Egenhofer *et al.* 1994), comprising the generalized regions, A^* and B^*, and their corresponding holes, A_H and B_H, together with the sixteen binary topological relations among these four regions (Figure 3). Eight of these sixteen binary topological relations are implied for any configuration between two single-holed regions, because each region must be *equal* to itself, A^* *contains* A_H, B^* *contains* B_H, and conversely A_H is *inside* A^*, and B_H is *inside* B^*.

	A^*	A_H	B^*	B_H
A^*	*equal*	*contains*	$t(A^*,B^*)$	$t(A^*,B_H)$
A_H	*inside*	*equal*	$t(A_H,B^*)$	$t(A_H,B_H)$
B^*	$t(B^*,A^*)$	$t(B^*,A_H)$	*equal*	*contains*
B_H	$t(B_H,A^*)$	$t(B_H,A_H)$	*inside*	*equal*

Fig. 3. Representation of the topological relation between two single-holed regions, A and B, comprising eight binary relations and their converses (the remaining eight relations are fixed for two single-holed regions)

At most the four relations $t(A^*,B^*)$, $t(A^*,B_H)$, $t(A_H,B^*)$, and $t(A_H,B_H)$ are required to specify any $t_{\mathrm{R_hR_h}}$ (Eqn. 1), because their converse relations are implied by the arc consistency constraint (Macworth 1977). These four relations are called the *constituent relations* of a topological relation between two single-holed relations. Their 2x2 matrix is a direct projection of the top-right elements of the spatial scene (Figure 3).

$$t_{\mathrm{R_hR_h}}(A,B) = \begin{bmatrix} t(A^*,B^*) & t(A^*,B_H) \\ t(A_H,B^*) & t(A_H,B_H) \end{bmatrix} \tag{1}$$

The *principal relation* between two single-holed regions, $\pi(t_{\mathrm{R_hR_h}})$, captures the relation between the two generalized regions. It is the top-left element of the constituent relations (Eqn. 2a). The other three constituent relations are referred to as the *inter-hole relation* $\omega(t_{\mathrm{R_hR_h}})$ (Eqn. 2b), the minor relation $\psi(t_{\mathrm{R_hR_h}})$ (Eqn. 2c), and the *reverse minor relation* $\psi^{-1}(t_{\mathrm{R_hR_h}})$ (Eqn. 2d).

$$\pi(t_{\mathrm{R_hR_h}}(A,B)) = t(A^*,B^*) \tag{2a}$$
$$\omega(t_{\mathrm{R_hR_h}}(A,B)) = t(A_H,B_H) \tag{2b}$$
$$\psi(t_{\mathrm{R_hR_h}}(A,B)) = t(A^*,B_H) \tag{2c}$$
$$\psi^{-1}(t_{\mathrm{R_hR_h}}(A,B)) = t(A_H,B^*) \tag{2d}$$

$$t_{1}\begin{bmatrix} d & d \\ d & d \end{bmatrix} \quad t_{2}\begin{bmatrix} m & d \\ d & d \end{bmatrix} \quad t_{3}\begin{bmatrix} o & d \\ d & d \end{bmatrix} \quad t_{4}\begin{bmatrix} o & d \\ m & d \end{bmatrix} \quad t_{5}\begin{bmatrix} o & d \\ d & d \end{bmatrix} \quad t_{6}\begin{bmatrix} o & d \\ cB & d \end{bmatrix} \quad t_{7}\begin{bmatrix} o & d \\ i & d \end{bmatrix} \quad t_{8}\begin{bmatrix} o & m \\ d & d \end{bmatrix}$$

$$t_{9}\begin{bmatrix} o & m \\ m & d \end{bmatrix} \quad t_{10}\begin{bmatrix} o & m \\ o & d \end{bmatrix} \quad t_{11}\begin{bmatrix} o & m \\ cB & d \end{bmatrix} \quad t_{12}\begin{bmatrix} o & m \\ i & d \end{bmatrix} \quad t_{13}\begin{bmatrix} o & o \\ d & d \end{bmatrix} \quad t_{14}\begin{bmatrix} o & o \\ m & d \end{bmatrix} \quad t_{15}\begin{bmatrix} o & o \\ o & d \end{bmatrix} \quad t_{16}\begin{bmatrix} o & o \\ o & m \end{bmatrix}$$

$$t_{17}\begin{bmatrix} o & o \\ o & o \end{bmatrix} \quad t_{18}\begin{bmatrix} o & o \\ cB & o \end{bmatrix} \quad t_{19}\begin{bmatrix} o & o \\ cB & m \end{bmatrix} \quad t_{20}\begin{bmatrix} o & o \\ cB & o \end{bmatrix} \quad t_{21}\begin{bmatrix} o & o \\ i & o \end{bmatrix} \quad t_{22}\begin{bmatrix} o & o \\ i & m \end{bmatrix} \quad t_{23}\begin{bmatrix} o & o \\ i & o \end{bmatrix} \quad t_{24}\begin{bmatrix} o & o \\ i & cB \end{bmatrix}$$

$$t_{25}\begin{bmatrix} o & o \\ i & i \end{bmatrix} \quad t_{26}\begin{bmatrix} o & cv \\ d & d \end{bmatrix} \quad t_{27}\begin{bmatrix} o & cv \\ m & d \end{bmatrix} \quad t_{28}\begin{bmatrix} o & cv \\ o & d \end{bmatrix} \quad t_{29}\begin{bmatrix} o & cv \\ o & m \end{bmatrix} \quad t_{30}\begin{bmatrix} o & cv \\ o & o \end{bmatrix} \quad t_{31}\begin{bmatrix} o & cv \\ cB & d \end{bmatrix} \quad t_{32}\begin{bmatrix} o & cv \\ cB & m \end{bmatrix}$$

$$t_{33}\begin{bmatrix} o & cv \\ cB & o \end{bmatrix} \quad t_{34}\begin{bmatrix} o & cv \\ i & d \end{bmatrix} \quad t_{35}\begin{bmatrix} o & cv \\ i & m \end{bmatrix} \quad t_{36}\begin{bmatrix} o & cv \\ i & o \end{bmatrix} \quad t_{37}\begin{bmatrix} o & cv \\ i & cB \end{bmatrix} \quad t_{38}\begin{bmatrix} o & cv \\ i & i \end{bmatrix} \quad t_{39}\begin{bmatrix} o & ct \\ d & d \end{bmatrix} \quad t_{40}\begin{bmatrix} o & ct \\ m & d \end{bmatrix}$$

$$t_{41}\begin{bmatrix} o & ct \\ o & d \end{bmatrix} \quad t_{42}\begin{bmatrix} o & ct \\ o & m \end{bmatrix} \quad t_{43}\begin{bmatrix} o & ct \\ o & o \end{bmatrix} \quad t_{44}\begin{bmatrix} o & ct \\ o & cv \end{bmatrix} \quad t_{45}\begin{bmatrix} o & ct \\ o & ct \end{bmatrix} \quad t_{46}\begin{bmatrix} o & ct \\ cB & d \end{bmatrix} \quad t_{47}\begin{bmatrix} o & ct \\ cB & m \end{bmatrix} \quad t_{48}\begin{bmatrix} o & ct \\ cB & o \end{bmatrix}$$

$$t_{49}\begin{bmatrix} o & ct \\ cB & cv \end{bmatrix} \quad t_{50}\begin{bmatrix} o & ct \\ cB & ct \end{bmatrix} \quad t_{51}\begin{bmatrix} o & ct \\ i & d \end{bmatrix} \quad t_{52}\begin{bmatrix} o & ct \\ i & m \end{bmatrix} \quad t_{53}\begin{bmatrix} o & ct \\ i & o \end{bmatrix} \quad t_{54}\begin{bmatrix} o & ct \\ i & cv \end{bmatrix} \quad t_{55}\begin{bmatrix} o & ct \\ i & cB \end{bmatrix} \quad t_{56}\begin{bmatrix} o & ct \\ i & ct \end{bmatrix}$$

$$t_{57}\begin{bmatrix} o & ct \\ i & i \end{bmatrix} \quad t_{58}\begin{bmatrix} o & ct \\ i & e \end{bmatrix} \quad t_{59}\begin{bmatrix} e & ct \\ i & d \end{bmatrix} \quad t_{60}\begin{bmatrix} e & ct \\ i & m \end{bmatrix} \quad t_{61}\begin{bmatrix} e & ct \\ i & e \end{bmatrix} \quad t_{62}\begin{bmatrix} e & ct \\ i & cB \end{bmatrix} \quad t_{63}\begin{bmatrix} e & ct \\ i & cv \end{bmatrix} \quad t_{64}\begin{bmatrix} e & ct \\ i & o \end{bmatrix}$$

$$t_{65}\begin{bmatrix} e & ct \\ i & ct \end{bmatrix} \quad t_{66}\begin{bmatrix} e & ct \\ i & i \end{bmatrix} \quad t_{67}\begin{bmatrix} cv & ct \\ d & d \end{bmatrix} \quad t_{68}\begin{bmatrix} cv & ct \\ m & d \end{bmatrix} \quad t_{69}\begin{bmatrix} cv & ct \\ o & d \end{bmatrix} \quad t_{70}\begin{bmatrix} cv & ct \\ o & m \end{bmatrix} \quad t_{71}\begin{bmatrix} cv & ct \\ o & o \end{bmatrix} \quad t_{72}\begin{bmatrix} cv & ct \\ o & cv \end{bmatrix}$$

$$t_{73}\begin{bmatrix} cv & ct \\ o & ct \end{bmatrix} \quad t_{74}\begin{bmatrix} cv & ct \\ cB & d \end{bmatrix} \quad t_{75}\begin{bmatrix} cv & ct \\ cB & m \end{bmatrix} \quad t_{76}\begin{bmatrix} cv & ct \\ cB & o \end{bmatrix} \quad t_{77}\begin{bmatrix} cv & ct \\ cB & cv \end{bmatrix} \quad t_{78}\begin{bmatrix} cv & ct \\ cB & ct \end{bmatrix} \quad t_{79}\begin{bmatrix} cv & ct \\ i & d \end{bmatrix} \quad t_{80}\begin{bmatrix} cv & ct \\ i & m \end{bmatrix}$$

$$t_{81}\begin{bmatrix} cv & ct \\ i & o \end{bmatrix} \quad t_{82}\begin{bmatrix} cv & ct \\ i & e \end{bmatrix} \quad t_{83}\begin{bmatrix} cv & ct \\ i & cv \end{bmatrix} \quad t_{84}\begin{bmatrix} cv & ct \\ i & cB \end{bmatrix} \quad t_{85}\begin{bmatrix} cv & ct \\ i & ct \end{bmatrix} \quad t_{86}\begin{bmatrix} cv & ct \\ i & i \end{bmatrix} \quad t_{87}\begin{bmatrix} i & d \\ i & d \end{bmatrix} \quad t_{88}\begin{bmatrix} i & m \\ i & d \end{bmatrix}$$

$$t_{89}\begin{bmatrix} i & o \\ i & d \end{bmatrix} \quad t_{90}\begin{bmatrix} i & o \\ i & m \end{bmatrix} \quad t_{91}\begin{bmatrix} i & o \\ i & o \end{bmatrix} \quad t_{92}\begin{bmatrix} i & o \\ i & cB \end{bmatrix} \quad t_{93}\begin{bmatrix} i & o \\ i & i \end{bmatrix} \quad t_{94}\begin{bmatrix} i & cv \\ i & d \end{bmatrix} \quad t_{95}\begin{bmatrix} i & cv \\ i & m \end{bmatrix} \quad t_{96}\begin{bmatrix} i & cv \\ i & o \end{bmatrix}$$

$$t_{97}\begin{bmatrix} i & cv \\ i & cB \end{bmatrix} \quad t_{98}\begin{bmatrix} i & cv \\ i & i \end{bmatrix} \quad t_{99}\begin{bmatrix} i & e \\ i & i \end{bmatrix} \quad t_{100}\begin{bmatrix} i & cB \\ i & i \end{bmatrix} \quad t_{101}\begin{bmatrix} i & ct \\ i & d \end{bmatrix} \quad t_{102}\begin{bmatrix} i & ct \\ i & m \end{bmatrix} \quad t_{103}\begin{bmatrix} i & ct \\ i & o \end{bmatrix} \quad t_{104}\begin{bmatrix} i & ct \\ i & e \end{bmatrix}$$

$$t_{105}\begin{bmatrix} i & ct \\ i & cv \end{bmatrix} \quad t_{106}\begin{bmatrix} i & ct \\ i & cB \end{bmatrix} \quad t_{107}\begin{bmatrix} i & ct \\ i & i \end{bmatrix} \quad t_{108}\begin{bmatrix} i & ct \\ i & ct \end{bmatrix} \quad t_{109}\begin{bmatrix} i & i \\ i & i \end{bmatrix} \quad t_{110}\begin{bmatrix} cB & d \\ i & d \end{bmatrix} \quad t_{111}\begin{bmatrix} cB & m \\ i & d \end{bmatrix} \quad t_{112}\begin{bmatrix} cB & o \\ i & d \end{bmatrix}$$

$$t_{113}\begin{bmatrix} cB & o \\ i & m \end{bmatrix} \quad t_{114}\begin{bmatrix} cB & o \\ i & o \end{bmatrix} \quad t_{115}\begin{bmatrix} cB & o \\ i & cB \end{bmatrix} \quad t_{116}\begin{bmatrix} cB & o \\ i & i \end{bmatrix} \quad t_{117}\begin{bmatrix} cB & cv \\ i & d \end{bmatrix} \quad t_{118}\begin{bmatrix} cB & cv \\ i & m \end{bmatrix} \quad t_{119}\begin{bmatrix} cB & cv \\ i & o \end{bmatrix} \quad t_{120}\begin{bmatrix} cB & cv \\ i & cB \end{bmatrix}$$

$$t_{121}\begin{bmatrix} cB & cv \\ i & i \end{bmatrix} \quad t_{122}\begin{bmatrix} cB & ct \\ i & d \end{bmatrix} \quad t_{123}\begin{bmatrix} cB & ct \\ i & m \end{bmatrix} \quad t_{124}\begin{bmatrix} cB & ct \\ i & o \end{bmatrix} \quad t_{125}\begin{bmatrix} cB & ct \\ i & e \end{bmatrix} \quad t_{126}\begin{bmatrix} cB & ct \\ i & cB \end{bmatrix} \quad t_{127}\begin{bmatrix} cB & ct \\ i & cv \end{bmatrix} \quad t_{128}\begin{bmatrix} cB & ct \\ i & ct \end{bmatrix}$$

$$t_{129}\begin{bmatrix} cB & ct \\ i & i \end{bmatrix} \quad t_{130}\begin{bmatrix} ct & ct \\ d & d \end{bmatrix} \quad t_{131}\begin{bmatrix} ct & ct \\ m & d \end{bmatrix} \quad t_{132}\begin{bmatrix} ct & ct \\ o & d \end{bmatrix} \quad t_{133}\begin{bmatrix} ct & ct \\ o & m \end{bmatrix} \quad t_{134}\begin{bmatrix} ct & ct \\ o & o \end{bmatrix} \quad t_{135}\begin{bmatrix} ct & ct \\ o & cv \end{bmatrix} \quad t_{136}\begin{bmatrix} ct & ct \\ o & ct \end{bmatrix}$$

$$t_{137}\begin{bmatrix} ct & ct \\ cB & d \end{bmatrix} \quad t_{138}\begin{bmatrix} ct & ct \\ cB & m \end{bmatrix} \quad t_{139}\begin{bmatrix} ct & ct \\ cB & o \end{bmatrix} \quad t_{140}\begin{bmatrix} ct & ct \\ cB & cv \end{bmatrix} \quad t_{141}\begin{bmatrix} ct & ct \\ cv & ct \end{bmatrix} \quad t_{142}\begin{bmatrix} ct & ct \\ cB & ct \end{bmatrix} \quad t_{143}\begin{bmatrix} ct & ct \\ e & ct \end{bmatrix} \quad t_{144}\begin{bmatrix} ct & ct \\ i & cv \end{bmatrix}$$

$$t_{145}\begin{bmatrix} ct & ct \\ i & cB \end{bmatrix} \quad t_{146}\begin{bmatrix} ct & ct \\ i & d \end{bmatrix} \quad t_{147}\begin{bmatrix} ct & ct \\ i & m \end{bmatrix} \quad t_{148}\begin{bmatrix} ct & ct \\ i & o \end{bmatrix} \quad t_{149}\begin{bmatrix} ct & ct \\ i & e \end{bmatrix} \quad t_{150}\begin{bmatrix} ct & ct \\ i & ct \end{bmatrix} \quad t_{151}\begin{bmatrix} ct & ct \\ i & i \end{bmatrix} \quad t_{152}\begin{bmatrix} ct & ct \\ ct & ct \end{bmatrix}$$

Fig. 4. The specifications of the 152 topological relations between two holed regions, captured by their constituent relations (*d* = *disjoint*, *m* = *meet*, *o* = *overlap*, *e* = *equal*, *cB* = *coveredBy*, *i* = *inside*, *cv* = *covers*, *ct* = *contains*)

While the domain of the constituent relations is the set of eight region-region relations, only a subset of the $8^4 = 4{,}096$ is feasible given the constraint that each hole must be included in its corresponding generalized region. For instance, the configuration with the principal relation *meet* and all other constituent relations *overlap* is impossible, whereas the configuration with the principal relation *meet* and all other constituent relations *disjoint* is feasible.

The complete set of feasible topological relations between two single-holed regions A and B is derived from the 4-region spatial scene (Figure 3). To specify a $t_{\mathrm{R_hR_h}}$, each of the four constituent relations is replaced in the spatial scene by a single t_{RR}. The $t_{\mathrm{R_hR_h}}$ is then feasible if (1) its 4-region scene is node-, arc-, and path consistent (Macworth 1977) and (2) there exists a planar graphical depiction of that spatial scene.

By taking into account the dependencies known from the 23 $t_{\mathrm{RR_h}}$ (Egenhofer and Vasardani 2007) the upper bound of 4,096 tests can be reduced. Splitting the *constituent* relations into two 1x2 matrices generates two $t_{\mathrm{RR_h}}$—one between simple region A^* and a single-holed region B (Eqn. 3a) and another between a simple region A_H and the single-holed region B (Eqn. 3b). Instead of four variables with a range of the eight t_{RR}, this approach leaves two variables that each range over the 23 $t_{\mathrm{RR_h}}$, yielding a total of $23^2 = 529$ possible combinations.

$$t_{\mathrm{RRh}}(A^*, B) = [t(A^*, B^*)\;\; t(A^*, B_H)] \tag{3a}$$
$$t_{\mathrm{RRh}}(A_H, B) = [t(A_H, B^*)\;\; t(A_H, B_H)] \tag{3b}$$

The set of all consistent configurations was derived with a scene consistency checker that iterated over the 23 feasible relations for each of the unknown t_{RR_h} and determined computationally whether that scene was node-consistent, arc-consistent, and path-consistent. Out of the 529 candidates, 152 combinations fulfilled these criteria. Their feasibility was confirmed by drawing for each, an example configuration (http://www.spatial.maine.edu/~mvasardani/152relations.pdf). Figure 4 provides the complete set of the 152 consistent relations and a naming scheme to enable the mapping from the graphical domain onto their symbolic representations. This naming scheme is purely symbolic as the numbers in the symbolic names do not imply an ordering. For instance, Figures 1c and 1d are examples of t_{17} and t_{81}, respectively, while Figures 2a-2e map onto t_1, t_{152}, t_2, t_{141}, and t_{143}. Like other sets of topological relations, the 152 $t_{\mathrm{R_hR_h}}$ are jointly exhaustive—there is always one of the 152 relations that applies between any possible pair of single-holed regions—and pairwise disjoint—for any configuration no more than one of the 152 relations applies.

4 Properties of the Single-Holed Region Relations

Are all four constituent relations needed to capture the 152 relations? To answer this question we examine the occurrence of the 152 relations for each constituent relation. Figure 5 summarizes the occurrences of all $t_{R_hR_h}$ by their principal relation $\pi(t_{\mathrm{R_hR_h}})$, inter-hole relation $\omega(t_{\mathrm{R_hR_h}})$, minor relation $\psi(t_{\mathrm{R_hR_h}})$, and reverse-minor relation $\psi^{-1}(t_{\mathrm{R_hR_h}})$. The counts of the latter two show the expected converse behavior, but otherwise the distributions of these counts differ widely.

rel	# ($\pi(t_{R_hR_h})$ = rel)	# ($\omega(t_{R_hR_h})$ = rel)	# ($\psi(t_{R_hR_h})$ = rel)	# ($\psi^{-1}(t_{R_hR_h})$ = rel)
disjoint	1	48	9	9
meet	1	22	7	7
overlap	56	23	23	23
equal	8	5	1	1
covers	20	12	23	1
inside	23	15	1	87
coveredBy	20	12	1	23
contains	23	15	87	1

Fig. 5. Counts (#) of $t_{R_hR_h}$ s by their principal relation $\pi(t_{R_hR_h})$, inter-hole relation $\omega(t_{R_hR_h})$, minor relation $\psi(t_{R_hR_h})$, and reverse-minor relation $\psi^{-1}(t_{R_hR_h})$

The summary reveals that most constraining principal relations are *disjoint* and *meet*, which each yield only a single possible relation for two single-holed regions. In both cases the relation between the generalized regions is so strong that the two holes must be *disjoint* and each hole must be *disjoint* from the other generalized region as well. For the remaining 150 relations, the mere knowledge of the principal relation leaves the $t_{R_hR_h}$ underdetermined, that is, one can exclude some—but not all—options for the three constituent relations with the holes. Among the underdetermined relations, those with the principal relation *equal* are the most constraining ones yielding eight variations, which is the same count as the cardinality of different region-region relations. This coincidence is not accidental, however, because when two generalized regions are *equal*, then they necessarily *contain* each other's holes, which between each other may assume any of the eight region-region relations. So any underdetermined $t_{R_hR_h}$ with an *equal* principal relation is completed with the specification of the inter-hole relation $\omega(t_{R_hR_h})$. Less restrictive is the principal relation *covers* (and its converse *coveredBy*). When A^* *covers* B^*, then it necessarily contains B_H, which means all but three of the 23 t_{RR_h} between A_H and a region with a hole B apply (A_H cannot be *equal* to, *cover* or *contain* B^*). Even less restrictive is the principal relation *contains* (and its converse *inside*). Since A^* *contains* B^* also implies A^* *contains* B_H, these $t_{R_hR_h}$ are completed with the information about which of the 23 t_{RR_h} holds between A_H and the region with hole *B*. The least constraining principal relation is *overlap*, for which 56 different cases apply.

The occurrences of $t_{R_hR_h}$ according to the inter-hole relation $\omega(t_{R_hR_h})$ show that no $t_{R_hR_h}$ is fully determined by its $\omega(t_{R_hR_h})$ alone, while the remaining two constituent relations—the minor $\psi(t_{R_hR_h})$ and the reverse-minor with the same distributions for pairs of converse relations—each has three uniquely defined relations: (1) when one generalized region coincides with the other region's hole and (2) when the generalized region is somehow contained in the region's hole, or converse the region's hole is somehow contained in the generalized region.

When considering combinations of the constituent relations, two of these duals exhibit good increases in uniquely defined relations: $\pi(t_{R_hR_h})$ and $\omega(t_{R_hR_h})$ together specify another 21 relations that are not yet covered by them individually, while $\psi(t_{R_hR_h})$ and $\psi^{-1}(t_{R_hR_h})$ together yield another 11 unique relations. The best result from considering triples— $\pi(t_{\mathrm{R_hR_h}})$, $\omega(t_{\mathrm{R_hR_h}})$, and $\psi(t_{\mathrm{R_hR_h}})$ or $\psi^{-1}(t_{\mathrm{R_hR_h}})$—yields altogether 66 uniquely specified relations. The analysis of the triples also shows that 20 of the 152 relations (i.e., 13%) are not covered by the union of all three-combination triples, therefore, requiring all four constituent relations.

The relation converse to t_{RhRh}, denoted by $\bar{t}_{\mathrm{RhRh}}$, is implied through the converse properties of each constituent relation (Eqns. 4a-d). For the matrix representation of t_{RhRh} (Eqn. 1) this means that $\bar{t}_{\mathrm{RhRh}}$ is the transposed matrix of the corresponding converse constituent relations (Eqn. 4e).

$$t(B^*,A^*) = \bar{t}(A^*,B^*) \tag{4a}$$

$$t(B^*,A_H) = \bar{t}(A_H,B^*) \tag{4b}$$

$$t(B_H,A^*) = \bar{t}(A^*,B_H) \tag{4c}$$

$$t(B_H,A_H) = \bar{t}(A_H,B_H) \tag{4d}$$

$$t_{\mathrm{RhRh}}(B,A) = \bar{t}_{\mathrm{RhRh}}(A,B) = \begin{bmatrix} \bar{t}(A^*,B^*) & \bar{t}(A^*,B_H) \\ \bar{t}(A_H,B^*) & \bar{t}(A_H,B_H) \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} t(B^*,A^*) & t(B^*,A_H) \\ t(B_H,A^*) & t(B_H,A_H) \end{bmatrix} \tag{4e}$$

This dependency allows us to derive for each of the 152 t_{RhRh} (Figure 4) its converse relation. For instance, t_{97} and t_{140} form a pair of converse relations, because the constituent relations along the main diagonal form converse pairs—*inside/contains* and *coveredBy/covers*—and the converses of the constituent relations off the diagonal map onto each other—*covers/coveredBy* and *inside/contains.*

A special role is assumed by those t_{RhRh} whose constituent relations are identical to their own converse relations, which identifies these t_{RhRh} as *symmetric* relations. An obvious symmetric relation is t_1 with each constituent relation *disjoint*, because the converses of all four constituent relations are *disjoint* again, whereas a less obvious case is t_{51} (with *overlap* and *disjoint* along the main diagonal being identical to their converses, while the two elements off the diagonal—*contains* and *inside*—map onto each other). Among the 152 t_{RhRh} eighteen relations are symmetric (t_1 through t_3, t_9, t_{15} through t_{17}, t_{31} through t_{33}, t_{51} through t_{53}, t_{58} through t_{61}, and t_{64}), while the remaining 134 relations form 67 pairs of converse relations.

5 Compositions Involving Single-Holed Region Relations

The primary method of inferring information about combinations of relations is their composition. If two relations t_i (A, B) and t_j (B, C) are known, then their composition over their common region B, denoted by t_i ; t_j , derives the set of possible relations $\{t_k\}(A, C)$. We are mostly interested in examining the influence of the hole on the inferences that result from a composition. It is suggested that when the hole of the common region of the two composed relations is taken into account, then the composition results comprise fewer relations, reducing ambiguities.

We consider the two composition cases that result in a t_{RhRh}: (1) the compositions t_{RhR} ; t_{RRh}—that is, the composition of a relation between a single-holed region and a region without a hole, with a relation between a region without a hole and a single-holed region—and (2) the compositions t_{RhRh}; t_{RhRh}—that is, the composition of two relations each between two single-holed regions. The results are also compared with the composition cases that result in a t_{RR} —the compositions t_{RR} ; t_{RR} and t_{RR_h} ; t_{R_hR}, both available from earlier investigations (Egenhofer 1994, Egenhofer and Vasardani 2007)—to examine the influence of the hole of the common region when the rest of the involved regions are plain.

It might be tempting to derive the compositions t_{RhRh}; t_{RhRh} simply from their constituent relations by applying the region-region composition (Egenhofer 1994) for each corresponding pair of relations and mapping these constituent compositions back onto $t_{R_hR_h}$. For instance, the composition of t_{109} ; t_{109} could be determined over each pair of constituent relations, deriving four times that *inside* ; *inside* = *inside*, so that t_{109} ; t_{109} is t_{109}. While this result is correct in this particular case, the approach is incomplete and too simplistic, leading at times to incorrect composition inferences. It would, for instance, derive for the composition of t_{99} ; t_{99} with three compositions of *inside* ; *inside* = *inside* and one composition of *equal* ; *equal* = *equal* the incorrect result t_{99}. Instead, the derivation of the compositions of t_{RhR} ; t_{RRh} and t_{RhRh} ; t_{RhRh} requires the more involved model of a spatial scene with the complete set of relations between generalized regions and their holes.

To derive the compositions of t_{RhR} ; t_{RRh}, a spatial scene over five regions (A^*, A_H, B, C^* and C_H) is needed as the framework for deriving the t_{RhR} ; t_{RRh} composition table. Regions *A* and *C* have a hole, whereas the common region *B* is simple, yielding 25 region-region relations (Figure 6). The four derived relations $t(A^*, C^*)$, $t(A^*, C_H)$, $t(A_H, C^*)$, and $t(A_H, C_H)$ are the constituent relations of the inferred t_{RhRh}.

	A^*	A_H	B	C^*	C_H
A^*	*equal*	*contains*	$t(A^*, B)$	$t(A^*, C^*)$	$t(A^*, C_H)$
A_H	*inside*	*equal*	$t(A_H, B)$	$t(A_H, C^*)$	$t(A_H, C_H)$
B	$t(B, A^*)$	$t(B, A_H)$	*equal*	$t(B, C^*)$	$t(B, C_H)$
C^*	$t(C^*, A^*)$	$t(C^*, A_H)$	$t(C^*, B)$	*equal*	*contains*
C_H	$t(C_H, A^*)$	$t(C_H, A_H)$	$t(C_H, B)$	*inside*	*equal*

Fig. 6. The spatial scene of five regions to derive the composition t_{RhR} (*A, B*) ; t_{RRh} (*B C*)

The range of $t(A, B)$ and $t(C, B)$ is the set of set of 23 t_{RhR}; therefore, there are $23^2 = 529$ compositions. The range of the inferred $t(A, C)$ is the set of 152 t_{RhRh}. The scene consistency checker found all compositions to be valid (i.e., node-, arc- and path-consistent so that no composition resulted in an empty relation). The two extreme scenarios—compositions with unique and universal results—reveal 44 unique cases (i.e., 8.3% of the 529 compositions) and six universal cases (i.e., 1.1%).

The analog approach was used to derive the t_{RhRh} ; t_{RhRh} composition table, starting with a spatial scene over six regions for three single-holed regions (Figure 7).

Since the range of each $t(A, B)$ and $t(B, C)$ is the 152 t_{RhRh}, there are $152^2 = 23{,}104$ compositions. The scene consistency checker found all of them valid—that is, each relation inferred with the three network consistency constraints has no empty relation as its composition result. Among the 23,104 compositions, 2,239 inferences (i.e., 9.7%) are unique and six (i.e., 0.03%) are universals.

	A^*	A_H	B^*	B_H	C^*	C_H
A^*	*equal*	*contains*	$t(A^*, B^*)$	$t(A^*, B_H)$	$t(A^*, C^*)$	$t(A^*, C_H)$
A_H	*inside*	*equal*	$t(A_H, B^*)$	$t(A_H, B_H)$	$t(A_H, C^*)$	$t(A_H, C_H)$
B^*	$t(B^*, A^*)$	$t(B^*, A_H)$	*equal*	*contains*	$t(B^*, C^*)$	$t(B^*, C_H)$
B_H	$t(B_H, A^*)$	$t(B_H, A_H)$	*inside*	*equal*	$t(B_H, C^*)$	$t(B_H, C_H)$
C^*	$t(C^*, A^*)$	$t(C^*, A_H)$	$t(C^*, B^*)$	$t(C^*, B_H)$	*equal*	*contains*
C_H	$t(C_H, A^*)$	$t(C_H, A_H)$	$t(C_H, B^*)$	$t(C_H, B_H)$	*inside*	*equal*

Fig. 7. The spatial scene of six regions to derive the composition t_{RhRh} (A, B) ; t_{RhRh} (B, C)

The t_{RhRh} ; t_{RhRh} composition table identifies that three t_{RhRh} relations are transitive (i.e., the composition of such relations results in the same relation). These are t_{61}, t_{109}, and t_{152}. Among them t_{61} is the *identity relation* for t_{RhRh}, because it is symmetric, transitive, and reflexive. For t_{61}, the pair of generalized regions A^* and B^*, as well as the pair of holes A_H and B_H, is *equal*—the two constituent relations along t_{61}'s main diagonal. These two constraints imply that A^* *contains* B_H and A_H is *inside* B^*, which are t_{61}'s two constituent relations off the main diagonal.

The typical non-transitive composition with a unique inference composes two different relations and implies a single relation of a different category. For instance, t_2 ; t_{152} implies t_1. Among the non-transitive compositions with unique inferences is, however, an interesting case that deviates from this pattern, providing a scenario that does not occur with region-region relations. The composition of t_{143} ; t_{143} a single-holed region whose hole is filled by another single-holed region, which in turn has a its hole filled with yet another single-holed region—is unique, but unlike a transitive relation, it does not result in t_{143}. Instead, this composition results in t_{152} (i.e., the most-inner nested region is fully contained in the hole of the outer most region).

6 Analysis of Compositions

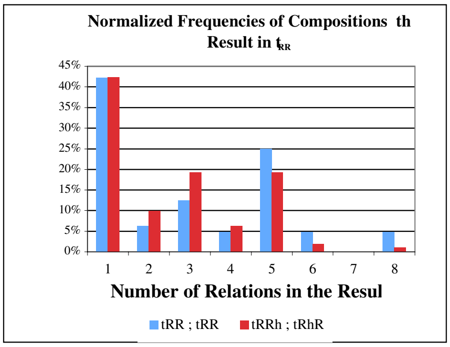
In order to examine a hole's influence on the inferences, we compare some properties of the composition tables involving holed regions. The results offer an insight into how the reasoning changes when holes are taken into account and justify the need for a separate qualitative model that acknowledges holed regions explicitly.

Four composition tables are available for this analysis. The first two composition tables (t_{RR} ; t_{RR} and $t_{\mathrm{RR_h}}$; $t_{\mathrm{R_hR}}$) both yield t_{RR}, while the second two composition tables ($t_{\mathrm{R_hR}}$; $t_{\mathrm{RR_h}}$ and $t_{\mathrm{R_hR_h}}$; $t_{\mathrm{R_hR_h}}$) both yield $t_{\mathrm{R_hR_h}}$. Therefore, each pair of comparisons has compatible domains. They also feature the same pattern of inserting a hole into the common region.

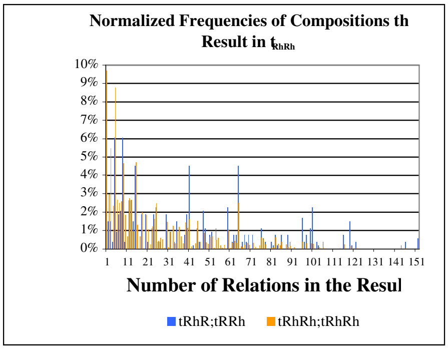
6.1 Absolute Frequencies of Compositions

The result of each composition consists of some subset of the complete set of relations over which the resulting type of relations ranges. The count of relations that are choices in the composition result is called the *composition cardinality*.

Composition cardinality of a unique composition is 1, a composition with two choices has composition cardinality 2, etc. Therefore, the upper bound of the composition cardinality applies to compositions that result in the universal relation, yielding a cardinality that equals to the number of relations in that set. For the compositions resulting in t_{RR}—t_{RR} ; t_{RR} and t_{RR_h} ; t_{R_hR}—the results are subsets of the eight t_{RR}, while for compositions resulting in $t_{R_hR_h}$—t_{R_hR} ; t_{RR_h} and $t_{R_hR_h}$; $t_{R_hR_h}$—the results are subsets of the 152 $t_{R_hR_h}$. The focus here is on observations on the frequencies of these subsets in each composition table. In order to make the results comparable, the frequencies of compositions are normalized (Figure 8).



(a)



(b)

Fig. 8. Normalized frequencies for compositions that result in (a) t_{RR} and (b) $t_{R_hR_h}$

Finding 1: *Inserting a hole, increases frequencies of crisper results.* Both normalized frequencies reveal that the insertion of a hole into the common region increases the frequencies of crisper compositions. This increase is due to the additional constraints that the hole puts on the relation with other regions. The frequencies of t_{RR} inferences increase for composition cardinalities 1–4 (Figure 8a), while for $t_{R_hR_h}$ inferences the frequencies increase, with a few exceptions, for composition cardinalities 1 to approximately 38.

Finding 2: *Compositions of relations with more holes make more fully unambiguous inferences.* For both compositions summarized in Figure 8, the compositions with the highest normalized frequencies are those with a singleton. For the two compositions with t_{RR} inferences, the normalized frequencies of the singleton results are very similar—42.2% for t_{RR} ; t_{RR} and 42.3% for t_{RR_h} ; t_{R_hR}—whereas for the two compositions with $t_{R_hR_h}$ inferences, the normalized frequencies increase—from 8.3% for t_{RR_h} ; t_{RR_h} to 9.7% for $t_{R_hR_h}$; $t_{R_hR_h}$. These quantified relative increases are smaller due to the larger range of the composition results (i.e., 152 vs. 8 possible relations). However, the larger number of single-holed regions involved in the compositions with $t_{R_hR_h}$ imposes more constraints on the reasoning, thereby increasing the number of fully unambiguous inferences.

6.2 Cumulative Frequencies of Compositions

The cumulative frequency of the composition cardinalities provides another measure for a hole's influence on the inferences. Each composition cardinality cc_i has a corresponding normalized frequency $\tilde{f}_i$. Composition cardinalities are ordered, such that $cc_i < cc_{i+1}$, with $cc_{\min} = 1$ and $cc_{\max}$ being the cardinality of the set's universal relation. Taken over the ordered sequence j of all composition cardinalities cc_j, the *cumulative frequency* of the composition cardinalities captures the sum of all normalized frequencies $\tilde{f}_1 \ldots \tilde{f}_j$. Since the composition cardinalities typically differ depending on the set of relations considered, they are also normalized onto a scale of 0 to 1 (by the cardinality of the set's universal relation), yielding *normalized composition cardinalities*. Figure 9 shows the cumulative normalized frequencies over normalized composition cardinalities for the two compositions t_{RR} ; t_{RR} and t_{R_hR} ; t_{RR_h}. Both are compositions over a single-holed region. The differences in the two graphs stem from properties of the underlying domains of the composition results—8 vs. 152 composition cardinalities—as well as the inference power of the different relation sets.

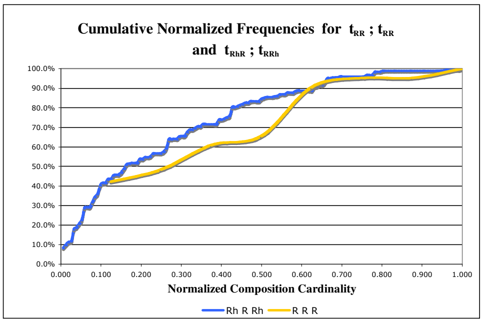


Fig. 9. Cumulative normalized frequencies over normalized composition cardinalities for t_{RR} ; t_{RR} and t_{R_hR} ; t_{RR_h}

Better comparisons can be made between pairs of compositions with equal inference ranges (Figure 10). Both pairs of corresponding curves have approximately the same origin and lead to the identical culmination point (at 100% accumulation for all composition cardinalities).

Finding 3: *The insertion of a hole increases monotonically the cumulative frequencies of the compositions, preserving the change in the accumulation pace*. The pairs of cumulative frequency graphs for t_{RR} ; t_{RR} and t_{RR_h} ; t_{R_hR} (Figure 10a) as well as t_{R_hR} ; t_{RR_h} and $t_{R_hR_h}$; $t_{R_hR_h}$ (Figure 10b) have similar shapes. In both cases, the curves increase strictly monotonically, except at their tail ends where they remain almost constant. The composition cardinalities over a single-holed region always have higher cumulative frequencies than their respective composition cardinalities over a simple region. The same shape and the higher cumulative frequencies, motivate further investigations into the properties of the cumulative normalized frequencies.

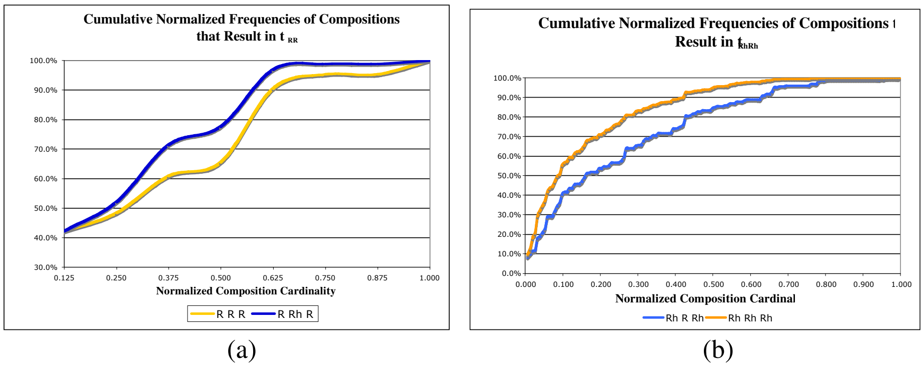


Fig. 10. Cumulative normalized frequencies over normalized composition cardinalities for compositions resulting in (a) t_{RR} and (b) $t_{\mathrm{R_hR_h}}$

Since the curves of corresponding composition pairs form essentially closed areas, it is feasible to quantify the difference in the cumulative frequencies by considering the increase in the area from the composition curve without a hole to the composition curve with a hole (Eqn. 5a and 5b).

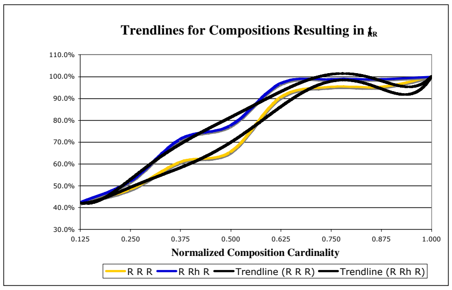
$$\Delta H_{RR} = \frac{\int_{\min RR_hR}^{\max RR_hR} RR_hR(x)\,dx}{\int_{\min RRR}^{\max RRR} RRR(x)\,dx} - 1 \tag{5a}$$

$$\Delta H_{R_hR_h} = \frac{\int_{\min R_hR_hR_h}^{\max R_hR_hR_h} R_hR_hR_h(x)\,dx}{\int_{\min R_hRR_h}^{\max R_hRR_h} R_hRR_h(x)\,dx} - 1 \tag{5b}$$

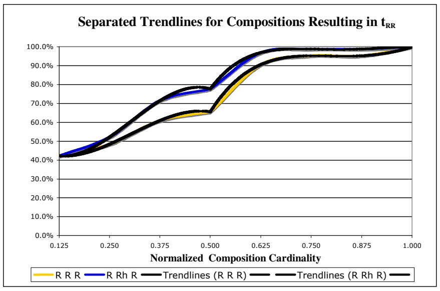
To calculate the area increases, 6th degree polynomial trend lines were fitted to the curves (Figure 11). Since the single trend lines for t_{RR} results did not generate the desired fit (Figure 11a), another approximation with the curves split in half at their apparent break points was used as well (Figure 11b). The increase calculated from these two approximations was the averaged. The quantitative values obtained are $\Delta H_{RR} = 8\%$ and $\Delta H_{R_hR_h} = 12\%$.

Finding 4: *Diversions from the average increase in cumulative frequencies reflect an increase of the frequencies of crisper composition results when a hole is inserted.* For the pair of compositions resulting in t_{RR}, the increase in the cumulative frequencies was assessed at the eight normalized composition cardinalities separately, in order to compare them with the average increase of 8%. For up to 0.5 normalized composition cardinalities, the increase in cumulative frequencies is above average—18.39% for up to 0.5 normalized composition cardinalities and 17.25% for up to 0.375 normalized composition cardinalities (which translates to 3 and 4 out of the eight t_{RR}). These numbers indicate that overall, when the composition's common region is a single-holed region, it increases the occurrence of crisper results, and specifically for composition results with up to 3 or 4 relations.

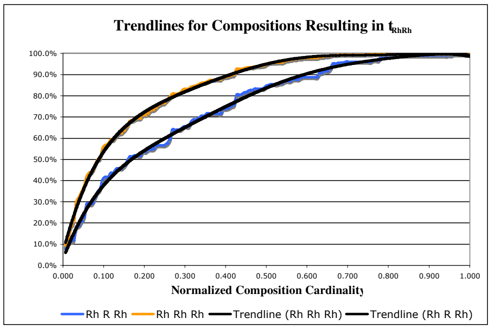
This increase of crisper composition results is also verified for compositions that result in $t_{R_hR_h}$. Samples of cumulative frequencies, taken at the normalized composition cardinalities of 0.125, 0.250, 0.375, 0.5, 0.625, 0.750, 0.875, and 1, show that the



(a)



(b)



(c)

Fig. 11. Approximations for cumulative frequency graphs: (a) and (b) two trendlines for compositions resulting in t_{RR} and (c) single trendline for compositions resulting in $t_{\mathrm{R_hR_h}}$

biggest increase in cumulative frequencies occurs up to 0.125 composition cardinalities—which translates to 19 out of 152 relations for the $t_{\mathrm{R_hR_h}}$ domain. This increase is 36%, in contrast to 12%, which was the average increase calculated by the difference in the areas under the trend lines. The increase in the cumulative frequencies continues to be higher than the average up to a normalized composition cardinality of 0.5 (i.e., compositions with 76 relations in their results). These observations verify the hypothesis that when the composition's common region is single-holed, crisper results are anticipated than for compositions over a region without a hole.

Finding 5: *Saturation of the cumulative frequency is reached earlier when the common region of the composition has a hole.* For $t_{\mathrm{RR_h}}$; $t_{\mathrm{R_hR}}$, for a composition cardinality of up to 0.5 relations the accumulation reaches 78%, compared to 66% reached at the same point for t_{RR} ; t_{RR}. Therefore, up through a composition cardinality of 4 relations, the accumulation of the $t_{\mathrm{RR_h}}$; $t_{\mathrm{R_hR}}$ is reached earlier than that of t_{RR} ; t_{RR}. For composition cardinalities of more than 0.5 normalized relations, the accumulation slows down for $t_{\mathrm{RR_h}}$; $t_{\mathrm{R_hR}}$, but increases for t_{RR} ; t_{RR}, confirming that the hole in the common region affects the reasoning by decreasing ambiguity of the inferences. Similarly due to the higher frequencies of crisper results, the cumulative frequencies for $t_{\mathrm{R_hR_h}}$; $t_{\mathrm{R_hR_h}}$ increase faster up to 0.5 normalized relations, after which they slow down. At this point, however, the pace increases for $t_{\mathrm{R_hR}}$; $t_{\mathrm{RR_h}}$.

Finding 6: *The accumulation pace is even faster when more holes are included in the pair of composed relations.* For the crisper composition results, the accumulation pace in $t_{R_hR_h}$; $t_{R_hR_h}$ is the highest among all composition scenarios considered. For up to 50% of the composition cardinalities, the cumulative normalized frequency has already reached 95%, in contrast to 66% for t_{RR} ; t_{RR}, 78% for t_{RR_h} ; t_{R_hR} and 84% for t_{R_hR} ; t_{RR_h}. When compared with the composition of the same domain approximately 60% of the accumulation has been reached already for a composition cardinality of up to 0.125 normalized relations, which translates to up to 19 relations per composition. The same percentage is reached only after the number of relations has more than doubled in the case of t_{R_hR} ; t_{RR_h}, where the cumulative frequency reaches 60% after the composition cardinality has reached 0.263 normalized relations (which translates to up to 40 relations per composition results). This faster pace continues until the accumulation is approximately 90%, after which the pace is smaller for $t_{R_hR_h}$; $t_{R_hR_h}$. The additional constraints that the bigger number of holes overall impose in this composition scenario (i.e., each pair of composed relations comprises of 2 single-holed regions whereas all the rest of the compositions happen between relations that comprise at most 1 single-holed region) are responsible for cripser composition results.

7 Conclusions

Holes in regions offer a plethora of topological relations, beyond those for regions without holes. While the amount of topological relations increase (from eight to 152 for two single-holed regions), the holes provide additional constraints for qualitative reasoning so that inferences may become crisper and, therefore, less ambiguous. The systematic analysis of four sets of compositions revealed that relations over holed regions yield more unique composition inferences, and reduce the ambiguities for close-unique inferences. The new quantitative measures enable the comparison of composition tables of relations over different domains (e.g., t_{RR}, t_{RR_h}, t_{R_hR}, and $t_{R_hR_h}$). They revealed that overall inferences for $t_{R_hR_h}$ are approximately 50% crisper than for other compositions with holed region-relations.

The results enable the investigation of a set of new questions. Do the 152 $t_{R_hR_h}$ form a relation algebra? Is their composition strong? What are tractable algorithms? Is the conceptual neighborhood graph for the 152 $t_{R_hR_h}$ planar? Given the large set of relations, it is of interest to derive coarser relations from this set, much like the generalization from RCC-8 to RCC-5.

Other future work items relate to the identification of those relations that maintain their inference power when regions have more than one hole, such as *disjoint* and *meet*, which survive any tampering with holes. What properties of the compositions change as holes are added, and what properties change when holes are removed, as typically applied in cartographic generalization? An approach in which holes have different levels of importance (e.g., by size of functionality) may overcome the limitations from all holes being the same.

Another set of new questions relate to the analogy of temporal and spatial reasoning. How similar are the compositions over 1-dimensional intervals with gaps to regions with holes? Do gapped interval relations increase at a comparable pace as holed region relations? Such insights would contribute to a better understanding of properties that are common to the spatial and temporal domains.

Acknowledgments

This work for partially supported by the National Geospatial-Intelligence Agency under grant number NMA201-01-1-2003 and a University of Maine Provost Graduate Fellowship.

References

Ahmed, N., Kanhere, S., Jha, S.: The Holes Problem in Wireless Sensor Networks: A Survey. Mobile Computing and Communications Review 9(2), 4–18 (2005)

Cassati, R., Varzi, A.: Holes and Other Superficialities. MIT Press, Cambridge (1994)

Cohn, A., Gotts, N.: The 'Egg-Yolk' Representation of Regions with Indeterminate Boundaries. In: Burrough, P., Frank, A. (eds.) Geographic Objects with Indeterminate Boundaries, pp. 171–187. Taylor & Francis, Bristol (1996)

Clementini, E., Di Felice, P.: An Algebraic Model for Spatial Objects with Indeterminate Boundaries. In: Burrough, P., Frank, A. (eds.) Geographic Objects with Indeterminate Boundaries, pp. 155–170. Taylor & Francis, Bristol (1996)

Egenhofer, M.: Deriving the Composition of Binary Topological Relations. Journal of Visual Languages and Computing 5(1), 133–149 (1994)

Egenhofer, M., Clementini, E., Di Felice, P.: Topological Relations Between Regions With Holes. International Journal of Geographical Information Systems 8(2), 129–144 (1994)

Egenhofer, M., Franzosa, R.: Point-Set Topological Spatial Relations. International Journal of Geographical Information Systems 5(2), 161–174 (1991)

Egenhofer, M., Herring, J.: 1994, Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases. Technical Report, Department of Surveying Engineering, University of Maine (1990)

Egenhofer, M., Vasardani, M.: Spatial Reasoning with a Hole. In: Winter, S., Duckham, M., Kulik, L., Kuipers, B. (eds.) COSIT 2007. LNCS, vol. 4736, pp. 303–320. Springer, Heidelberg (2007)

Lewis, D., Lewis, S.: Holes. Australasian Journal of Philosophy 48, 206–212 (1970)

Li, S., Ying, M.: Region Connection Calculus: Its Models and Composition Table. Artificial Intelligence 145(1-2), 121–146 (2003)

Mackworth, A.: Consistency in Networks of Relations. Artificial Intelligence 8(1), 99–118 (1977)

Papadias, D., Theodoridis, Y., Selis, T., Egenhofer, M.: Topological Relations in the World of Minimum Bounding Rectangles: A Study with R-trees. SIGMOD Record 24(2), 92–103 (1995)

Randell, D., Cohn, A., Cui, Z.: A Spatial Logic Based on Regions and Connection. In: Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning, pp. 165–176. Morgan Kaufmann, San Mateo (1992)

Schneider, M., Behr, T.: Topological Relationships between Complex Spatial Objects. ACM Transactions on Database Systems 31(1), 39–81 (2006)

Stefanidis, A., Nittel, S.: Geosensor Networks. CRC Press, Boca Raton (2004)

Varzi, A.: Reasoning about Space: The Hole Story. Logic and Logical Philosophy 4, 3–39 (1996)

Validation and Storage of Polyhedra through Constrained Delaunay Tetrahedralization

Edward Verbree[1] and Hang Si[2]

[1] Delft University of Technology
Research Institute OTB, Section GIS-technology
Jaffalaan 9, 2628BX Delft, the Netherlands
e.verbree@tudelft.nl
[2] Weierstrass Institute for Applied Analysis and Stochastics
Research Group: Numerical Mathematics and Scientific Computing
Mohrenstr. 39, 10117 Berlin, Germany
si@wias-berlin.de

Abstract. Closed, watertight, 3D geometries are represented by polyhedra. Current data models define these polyhedra basically as a set of polygons, leaving the test on intersecting polygons or open gaps to external validation rules. If this testing is not performed well, or not at all, non-valid polyhedra could be stored in geo-databases. This paper proposes the utilization of the Constrained Delaunay Tetrahedralization (CDT) for the validation (i.e. check on self-intersecting and closeness) of polyhedra on the one hand, and the efficient storage of valid polyhedra on the other hand. The paper stresses on the decomposition of a polyhedron through a CDT and the possibility to store and compose the polyhedron through the vertices of the CDT, a bitmap that indicates which faces of the Delaunay Tetrahedralization (DT) links to a CDT-face, and a list of non-recovered CDT-faces.

1 Introduction

Real world objects are characterized by a particular representation, identified and captured, and stored according a specified datamodel in a geo-database. For applications within the 3D domain, one cannot work any longer with ‘down to earth’ flattened objects, which are defined as polygon footprints attached to a 2D or 2.5D surface. An appropriate 3D representation is needed, which has to be supported by a suitable 3D data model.

The polyhedral approach, as described in [1] defines the boundary of a 3D primitive (or simple object) as a set of polygons, where the vertices of each polygon are co-planar and valid according the Simple Feature Specification of the OGC [2], i.e. non self-intersecting and closed. The polygons, defining the boundary of the 3D primitive, should be connected to each other in such a way that the 3D primitive itself is ‘closed’ (or ‘watertight’). That yields the set of polygons consist of non mutual-intersecting polygons and does not leaves open gaps, resulting in a connected interior for the 3D primitive (simple object).

1.1 Data Models for Polyhedra

Three-dimensional geometries (simple solids) have to be stored in a geo-database according to a supporting data model. In [3] a review is given of the ISO 19107 Schema [4], OGC GML specification [5] and Oracle Spatial SDO_GEOMETRY data type to store 3D geometries. This paper [3] defines more specific and refined rules for valid geometries. In relation to standardization organizations it states: “ISO and OGC have tried to give unambiguous and complete definitions of valid geometric primitives. But as it was already pointed out in [6] it turns out that the standards are not unambiguous and complete, even in the case of 2D-polygons. For 3D geometric primitives there is an abstract ISO specification [4], but this is not the needed implementation specification”.

The GML and ISO definition for solids reads: “A solid is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces (shells). A shell is represented by a composite surface, where every shell is used to represent a single connected component of the boundary of a solid. It consists of a composite surface (a list of orientable surfaces) connected in a topological cycle. Unlike a ring, a shell’s elements have no natural sort order. Like rings, shells are simple. The element ‘exterior’ specifies the outer boundary of the solid. Boundaries of solids are similar to surface boundaries. In normal 3-dimensional Euclidean space, one (composite) surface is distinguished as the exterior. The element ‘interior’ specifies the inner boundary of the solid.”

In [3] a description of Simple Solids (Polyhedra) in Oracle is given: “A simple solid is defined as a ‘Single Volume’ bounded on the exterior by one exterior composite surface and on the interior by zero or more interior composite surfaces. To demarcate the interior of the solid from the exterior, the polygons of the boundary are oriented such that their normal vector always point “outwards” from the solid.”

The Computational Geometry Algorithms Library (CGAL) [7] gives the following work-definition of a polyhedron: “Polyhedral surfaces in three dimensions are composed of vertices, edges, facets and an incidence relationship on them. The organization beneath is a halfedge data structure, which restricts the class of representable surfaces to orientable 2-manifolds - with and without boundary. If the surface is closed we call it a polyhedron. Each edge is represented by two halfedges with opposite orientations. Facets are defined by the circular sequence of halfedges along their boundary. The halfedges along the boundary of a hole are called border halfedges and have no incident facet. An edge is a border edge if one of its halfedges is a border halfedge. A surface is closed if it contains no border halfedges. A closed surface is a boundary representation for polyhedra in three dimensions.”

1.2 Validation of Polyhedra

Surprisingly, existing schemas, specifications and even implementations define datamodels to store 3D geometries, but they are not limited to store valid, thus ‘watertight’, 3D geometries only: also ‘non-watertight’ (boundary intersecting, open, non-connected interior) 3D geometries can be stored. In [3] the following validation rules/tests of solids are defined. These tests check on both a valid, closed boundary as on the connectedness of the interior.

These operations should use tolerance values to deal with the problems to express exact fractions in decimal notion with a finite number of binary digits.

- Single Volume check: the volume should be contiguous:
  - Closed test: the boundary has to be closed.
  - Connect test: the volume has to be connected. This means each component of the solid should be reachable from any other component.
- Inner-outer check:
  - Every surface marked as an inner boundary should be 'inside' the solid defined by the exterior boundary.
  - Inner boundaries may never intersect but only touch under the condition that the interior of the solid remains connected.
- Orientation check: The volume bounded by the exterior boundary is computed as a positive value if every face is oriented such that each normal is pointed away from the solid due to the Greens Theorem. Similarly, the volume bounded by the interior boundary is computed as a negative value. If each exterior and interior boundary obeys this rule and they pass the connect test as well, then this check is passed.
- Element-check: Every specified surface is a valid surface.

A possible way to avoid non-valid polyhedra, i.e. self-intersecting faces (a non circular sequence of halfedges) or non-planar faces, is to restrict the boundary of the polyhedra to be composed of a set of triangular facets. But even then this 'solid object' has to be validated, as it is still possible that the triangular facets of this 'polyhedron' could intersect each other.

1.3 Storage of Polyhedra through Tetrahedralization

The idea to use a tetrahedralization to represent polyhedra is based on previous work by the first author in representing Planar Maps through Conforming Delaunay Triangulations [8]. A method was described to store (encode) and receive (decode) a Planar Map (PM) through a Constrained Delaunay Triangulation (CDT) with applications in a server-client environment. Planar maps are embeddings of topological maps into the plane. A planar map subdivides the plane into vertices, edges, and faces [9]. In two dimensions a true, conforming, Delaunay Triangulation that constrains to the input can be created by adding Steiner points at the edges of the Planar Map.

To store a PM the server creates a CDT of the edges of the PM. As the PM is now embedded by the CDT it is sufficient to send to the client the list of coordinates of the CDT nodes and an efficient encoded bitmap of the corresponding PM-CDT edges. The client determines a Delaunay Triangulation (DT) of the received list of coordinates of the CDT nodes. The DT at the client side is - omission degenerated cases - equivalent to the CDT at the server side. The edges of the PM are recovered within this DT by the bitmap (true/false) of the corresponding PM-CDT edges.

Within [8] it was stated: "Despite all these considerations, the encoding (decomposition) of Planar Maps by Conforming Delaunay Triangulations could be extended to the third dimension. Polyhedron boundary representations could be encoded (decomposed) and decoded (composed) through a conformal tetrahedralization". That idea was presented in [10]. But, although polyhedra can be represented by a Conforming Delaunay Tetrahedralization, this can result in an extraordinary amount of added Steiner Points. In two-dimensions [11] shows an example that two-dimensional conforming Delaunay triangulations may need a quadratic number of Steiner points with respect to the input number of nodes. Therefore, we have adapted the method in this paper by applying Constrained Delaunay Tetrahedralization.

This paper links up with research by Penninga at al. on a simplicial complex-based DBMS approach to 3D topographic data modeling' [12] that has a focus on updating features in a TEN-based DBMS approach [13].

1.4 Our Work

We propose the use of a Constrained Delaunay Tetrahedralization (CDT) to validate and store polyhedra. By definition a polyhedron has to have one closed, watertight, polygonal outer surface and possible one or more inner surfaces. The validation of this important property has to be done in advance, as most geo-databases allow the storage of polygonal surfaces and polyhedra through the same datamodel. To check whether or not a polygonal surface is closed, and thus watertight, is not a straightforward task, as even the existing standards and specifications defining polyhedra (simple solids) are ambiguous.

The motivation behind the use of a CDT is given by its construction scheme. First and for all, if a polyhedron is valid, the faces of the derived CDT have to match the faces of its surface triangulation completely. This requirement holds also for polygonal surfaces, but as the final phase of the tetrahedralization removes the tetrahedra outside the surface triangulation, only closed, watertight, polyhedra survive.

The second advantage of applying a CDT is given by the possibility to store the geometry of the polyhedron by only its nodes, the added Steiner points, a Delaunay Tetrahedralization (DT) of these nodes and Steiner points, a bitmap of the lexical ordered faces (natural sorted on three nodes) of this DT (thus only one bit per face), and a list of the DT missing triangles of its surface triangulation. The efficiency of this approach depends on one hand on the amount of added Steiner points and on the other hand on how much the CDT conforms to the DT.

1.5 Outline

In the preliminary section of this paper (section 2) we describe the possibilities and limitations of current datamodels for the storage and validation of polyhedra. We give formal definitions of Conforming Delaunay Tetrahedralization and Constrained Delaunay Tetrahedralization. In section 3 we focus on the issue of validating polyhedra by means of CDT and we discuss the possibility to detect self-intersecting polygons and the distinction between (closed, watertight) polyhedra and polygonal surfaces. Section 4 describes the possibility to store polyhedra through CDT and some examples and test results on the efficiency of this approach are given. Section 5 resumes with the conclusions.

2 Preliminaries

This section presents the essential definitions and notations of the geometric objects used in this paper. These objects are based on the fundamental concepts developed in topology and geometry. Good introductory texts are given by Edelsbrunner [14] and Ziegler [15].

2.1 Polyhedra and Faces

In this section, we define a general and therefore not necessarily convex polyhedron and its faces.

Definition 1 (Polyhedron). A *polyhedron* P in R^d is the union of a finite set P of convex polyhedra, i.e., $P=\cup_{U\in P}U$, and the space of P is connected.

The *dimension* dim(P) is the largest dimension of a convex polyhedron in P. Note that P may contains holes in its interior. Whatever, we require that the space of P must be connected, i.e. any two points in the interior of P can be connected through a path in the interior of P. See Fig. 1 for examples.

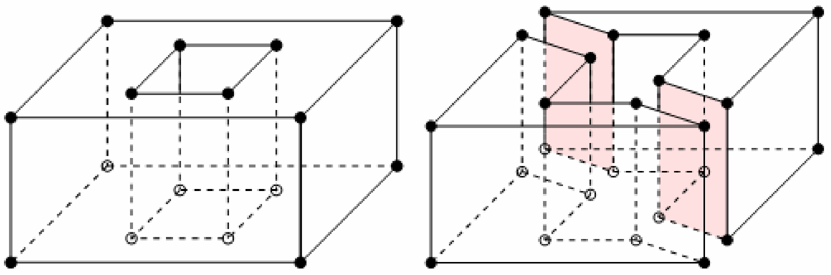


Fig. 1. Polyhedra and faces. Left: A three-dimensional polyhedron (a torus) formed by the union of four convex polytopes. It consists of 16 vertices (zero-faces), 24 edges (1-faces), 10 two-faces (the faces at top and bottom are not simply connected), and 1 three-face (which is itself). Right: Two three-dimensional polyhedra. Each one has 12 vertices, 18 edges, 8 two-faces, and 1 three-faces. The shaded area highlights two 2-faces whose points have the same face figures.

There are few definitions about faces of a non-convex polyhedron. The following definitions of faces are mainly by [14] with only difference in the connectness of the faces. Let $B(\mathbf{x},r)$ denote the open ball in R^d centered at $\mathbf{x}$ with radius r.

For a point $\mathbf{x}$ in a polyhedron P we consider a sufficiently small neighborhood $N_\varepsilon(\mathbf{x})=(\mathbf{x}+B(\mathbf{0},\varepsilon))\cap P$. The *face figure* of $\mathbf{x}$ is the enlarged version of this neighborhood within P, i.e., $\mathbf{x}+\cup_{\lambda>0}\lambda(N_\varepsilon(\mathbf{x})-\mathbf{x})$.

Definition 2 (Face). A *face F* of a polyhedron P is the closure of a maximal connected set of points with identical face figures.

By this definition, a face of P may contain holes in its interior, but it is always connected. See Fig. 1 for examples.

A face F of P is again a polyhedron. Particularly, $\emptyset$ is a face of P. If all convex polyhedra in P have the same dimension, then P itself is a face of P. All other faces of P are *proper* faces of P. We also write $F \leq P$ or $F < P$ if F is a face or a proper face of P. The faces of dimension 0, 1, $dim(P)-2$, and $dim(P)-1$ are called *vertices*, *edges*, *ridges*, and *facets*, respectively. The set of all vertices of P, the *vertex set*, will be denoted by $vert(P)$. The set of all proper faces of P is called the *boundary complex* of P, denoted as $bd(P)$. The *interior* $int(P)$ is $P-bd(P)$.

Note that a polyhedron may be unbounded. In the scope of this work, we always assume that a polyhedron is bounded, i.e., it is a polytope. In our later discussions, a polytope can be convex or non-convex.

2.2 Delaunay Triangulation / Tetrahedralization

Let S be a finite set of points in R^d. A *Delaunay triangulation* [16] (abbreviated as DT) of S is a triangulation T such that *(i)* the vertex set of T is S, *(ii)* the convex hull of S equals to the underlying space of T, and *(iii)* (Delaunay criterion) every simplex (i.e. triangle in 2D-space, tetrahedron in 3D-space) of T has a circumscribed ball whose interior contains no points of S. **Fig. 2** illustrates a two-dimensional Delaunay triangulation. The DT of a three-dimensional point set is also called a *Delaunay tetrahedralization*.

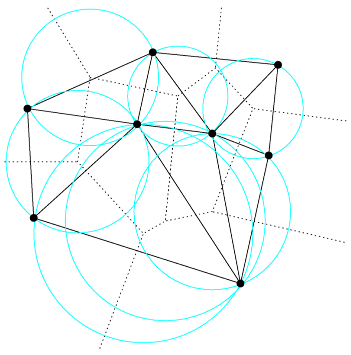


Fig. 2. The Delaunay triangulation (shown in solid black lines) of a two-dimensional point set. All circumcirles (shown in blue) of the triangles are empty. The dotted lines show the dual Voronoi diagram.

2.3 Conforming Delaunay Triangulation

Let P be a polygon. The DT of the vertices of P does not necessarily contain all the boundaries of P. A *conforming Delaunay triangulation* T of P is a Delaunay triangulation, such that

(i) each vertex of *P* is a vertex of *T*,
(ii) every boundary of *P* is a union of simplices of *T*, and
(iii) every simplex of *T* satisfies the Delaunay criterion.

Usually, a conforming DT of *P* will contain some additional points (so-called *Steiner points*) which do not belong to *P* (See Fig. 3 left for an example). It is worth to mention, even for a simple polygon, a large number of Steiner points may be needed. Let *n* be the number of points in the polygon. A simple example which needs $\Omega(n^2)$ Steiner points is given in [11].

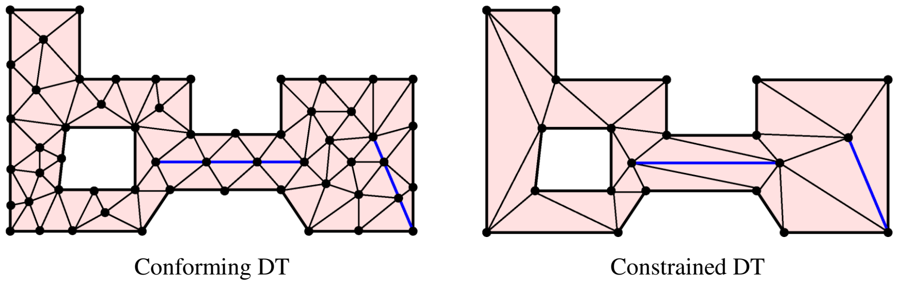


Fig. 3. Left: A conforming Delaunay triangulation of a two-dimensional Polygon P. Right: A constrained Delaunay triangulation of P.

2.4 Constrained Delaunay Triangulation

An alternative approach to form a triangulation of a polygon *P* is to use a Delaunay-like triangulation (not necessary to be a DT) which respects the boundaries of *P*.

The visibility between two vertices $\mathbf{p},\mathbf{q} \in P$ is defined as follows: **p** and **q** are *invisible* to each other if the line segment **pq** intersects with any boundary of *P* at an interior point of **pq**.

Let *T* be a triangulation such that the underlying space of *T* is the convex hull of the vertices of *P*. A simplex $\sigma \in T$ is *constrained Delaunay* if there exists a circum circle B_σ of σ such that either B_σ contains no other vertices of *T* or there is no vertex inside B_σ which is visible from the interior of σ. Then *T* is a *constrained Delaunay triangulation* (abbreviated as CDT) of *P* if

(i) Every boundary of *P* is represented by a union of simplices of T.
(ii) Every simplex of T is constrained Delaunay.

A two-dimensional CDT is illustrated in Fig. 3 at the right. A three-dimensional CDT is also called a *constrained Delaunay tetrahedralization.*

The above definition implies that a CDT of *P* may contain Steiner points, i.e., points which do not belong to *P*. When it is the case, we call it a *Steiner* CDT. Otherwise, it is called a *pure* CDT. It is well known that a pure CDT of a polyhedron may not exist. For instance, given the Schönhardt polyhedron [18], the problem to decide whether a pure CDT of *P* exists or not, is NP-complete [19]. On the other hand, there are infinitely many Steiner CDTs of *P*.

As in the 2-Dimensional case, many Steiner points can be needed to obtain a *genuine* Conforming Delaunay Tetrahedralization, see i.e. [20], [21], [22].

If Steiner points are allowed, Chazelle [23] showed that any simple polyhedron of n vertices may need $O(n^2)$ Steiner points, and this bound is tight in the worst case. Chazelle and Palios [24] presented an algorithm to decompose a simple polyhedron using $O(n+r^2)$ Steiner points, where r is the number of reflex edges (a quantitative measure of nonconvexity) of P . However, even for a simply shaped polyhedron, i.e. Fig. 4 (top-left) this algorithm will introduce unnecessarily large number of Steiner points, see Fig. 4 (top-right). More practical approaches using conforming Delaunay triangulations [20], [21] and constrained Delaunay triangulations [25], [26] are proposed, see Fig. 4 (bottom-left) and (bottom-right). However, no polynomial upper bound on the number of Steiner points is known; see the 22nd open problem in [14].

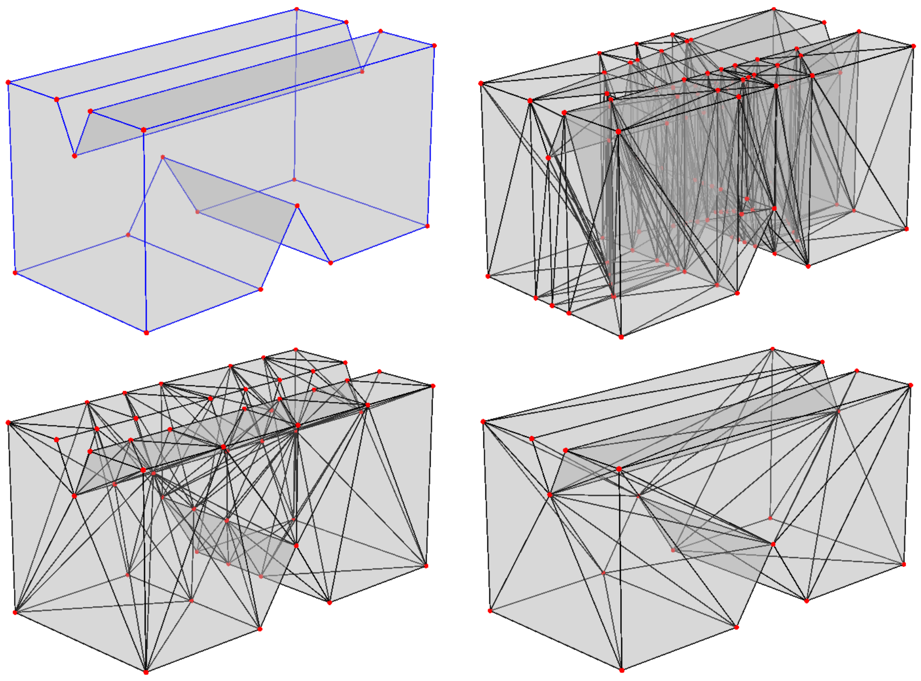


Fig. 4. Comparison of various approaches in decomposition (tetrahedralization) of a polyhedron: A ‘simple’ polyhedron with 20 vertices and 2 reflex can end up with 51 vertices and 103 tetrahedra to create a Conforming Delaunay Tetrahedralization

3 Validation of Polyhedra through CDTs

The data models presented in Section 1.1 do have one feature in common: the parts (i.e. polygons, shells, facets) of a three-dimensional solid are given. However there is no guarantee these parts are bound together to form a valid polyhedron. Various approaches have been proposed for validating a polyhedron from a given representation, see e.g., [3]. In this section, we propose a new approach for validating a polyhedron.

3.1 Decomposition of Polyhedra through Tetrahedralization

Our approach is based on this simple observation: A valid polyhedron *P* can be decomposed into a tetrahedralization *T* such that the boundary of *P* is represented as a union of simplices of *T*. This observation can be easily proved: Since *P* is valid, then let *T* be a Constrained Delaunay Tetradedralization (CDT) of *P*. Otherwise, if such a *T* does not exist, then either the boundary *bd(P)* of *P* contains self-intersections or *bd(P)* is not closed. Hence, the problem of validating a polyhedron *P* can be transformed into the problem of finding a tetrahedralization of *P*. This is a long studied problem in computational geometry (see e.g., [23], [24], [27]), as well as in mesh generation (see e.g. [28], [29], [30]).

To validate *P* takes two steps: *(1)* check the self-intersection in the boundary of *P*, and *(2)* generate the CDT of *P*. Note that step *(2)* can be called only if step *(1)* is successful. Then *P* is valid only if a CDT of *P* can be generated, the boundary of *P* is represented as a union of simplices of *T*, and the interior of *T* is connected.

The generation of CDTs is an active research topic. The discussion of such methods is out of the scope of this work, we refer to the research paper of Shewchuk [25] and Si et al [26]. A software implementation of the CDT algorithms can be found in the program `TetGen` [31]. In the following, we briefly discuss the approach to detect self-intersections.

3.2 Self-intersection Detect

A basic requirement for a 3D polyhedron *P* is that any two facets of *P* should touch only along their common faces. Let T_1 and T_2 be two triangles in R^3, let $U=T_1 \cap T_2$. We say that T_1 and T_2 *intersect* each other if $U \neq \emptyset$ and U is not a proper face of both T_1 and T_2. See Fig. 5 for examples.

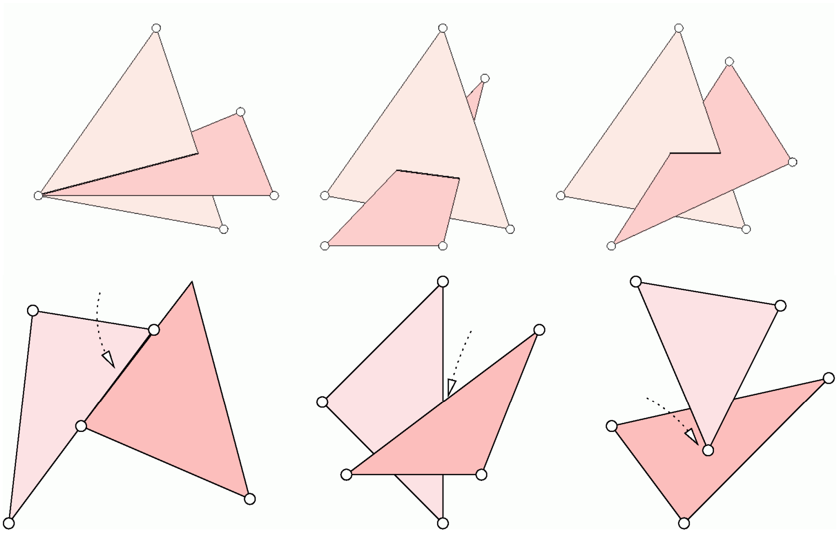


Fig. 5. Possible cases of two triangles in R^3 can intersect each other.

The problem. Consider the set of triangles forming the triangulation F of the boundary $bd(P)$ of P, find all intersected pairs of triangles in F.

The primitive operation for this task is a function $\textsc{Tri\_Tri\_Intersect}(T_1,T2)$. That is, it takes two triangles T_1,T_2 in R^3 as inputs, return a value indicating whether T_1 and T_2 intersect each other or not.

Fast algorithms are proposed to test the intersection between three-dimensional triangles, such as Möller [32] and Guigue et al [33]. In practice, Möller's algorithm is less robust since it needs to compute an intermediate line interval which requires arithmetic precision. Guigue et al's algorithm relies exclusively on orientation predicates, which eliminates the intermediate error caused by floating-point arithmetics. However, both algorithms do not satisfy our goal. Let $U=T_1 \cap T_2$. The problem lies that both algorithm only detect whether U is empty or not. In our problem, we need to distinguish more cases when $U \neq \emptyset$. To distinguish these cases needs to handle all degenerate cases.

We implemented a triangle-triangle test algorithm. The idea is similar to that of Guigue et al [33], only the three-dimensional orientation test is involved. It further classifies all degenerate cases based on the study of the signs. A trivial approach to check the boundary self intersection of the polyhedron is just to test the intersection of triangles pair by pair, which takes $O(m^2)$ time, where m is the number of triangles. A simple way to improve the speed is to filter out triangle pairs that can not intersect by first doing the intersection of their bounding boxes whose sides are parallel to the axes. It has been shown [34] that such box intersection can be reported in $O(m\log^2 m+J)$ time, where J is the number of intersected pairs.

We implemented a divide-and-conquer approach for reducing the number of intersection tests. Starting from the bounding box of the vertices, it recursively partitions the box into smaller boxes, until the number of triangles in a box is not decreased anymore. Then a brute-force pairwise triangle-triangle intersection test is performed. This approach runs in time $O(m\log m+I^2)$, where I is the largest number of triangles appearing in a box which is not partitioned. It is possible that $I=m$, i.e., the worst-case time complexity is $O(m^2)$. For most inputs which have a dense point set, I is relatively small comparing to m. Fig. 6 illustrates an example.

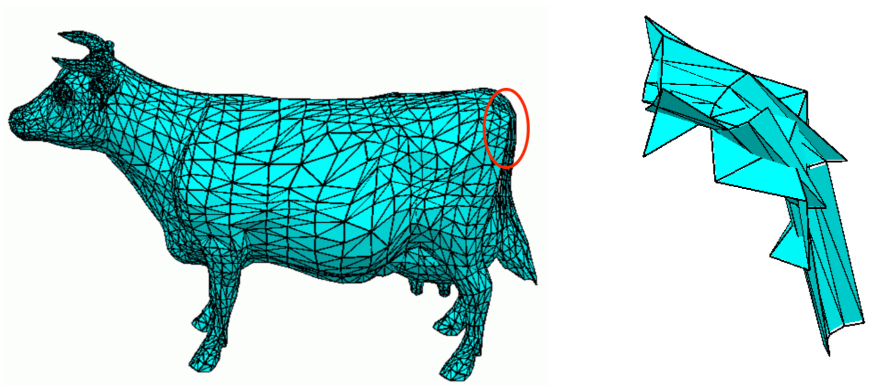


Fig. 6. Boundary intersection check. A surface mesh of a Cow is shown left. The triangles on the tail (the annotated place) are intersecting each other shown on the right.

4 Storage of Polyhedra through Constrained Delaunay Tetrahedralization

To proof the efficiency (i.e. gain of storage) of decomposition (encoding) and composition (decoding) a polyhedron through means of a tetrahedralization is not trivial.

The basic idea of this method is the decomposition of a given polyhedron by a Constrained Delaunay Tetrahedralization (CDT) of the nodes of the polyhedron. As shown in section 2, extra Steiner points are needed to make sure each polygon of the polyhedron is represented by a set of faces of the CDT. Within the obtained CDT, the faces which 'belong' to the polyhedron are marked with a binary bit '1', and the faces of the CDT that do not 'belong' to the polyhedron (thus faces inside or outside the polyhedron) are marked with a binary bit '0'. The composition is performed by a regular Delaunay Tetrahedralization (DT) of the original nodes of the given polyhedron plus the extra Steiner points.

4.1 Implementation Issues

One complication within this process is that the set of faces of the DT created during the composition phase can be different of the set of faces of the CDT created during the decomposition phase. The first reason for this is because the Constrained DT (CDT) does not have to be a Conforming Constrained Delaunay Tetrahedralization (CCDT): a CDT-tetrahedron does not have to obey in general to the Delaunay-criterion of an empty circumsphere. One option to work around that problem is to create during the decomposition phase a CCDT also, as suggested in [10]. But in creating a CCDT one needs to add far more Steiner points [20], [21], [22] and this extra amount of Steiner points limits this method. The second reason is caused by possible degenerated cases: five or more points directly on one circumsphere can be tetrahedronized in a different way by another DT-algorithm. So, even if a CCDT can be created, the DT of the same set of input points can result in another set of tetrahedra. One possibility to 'solve' this problem is to define a unique decision rule, i.e. two preferred directions as presented in for 2-dimensional DT in [35]. This method could be extended to a Delaunay Tetrahedralization, but this will cause some issues in defining directions in 3D.

For this reason, the coding of the faces 'belonging' to the polyhedron is based on the DT. One binary bit is needed for each face of the DT. A DT-face is to be marked (flagged) with a binary bit '1' when the corresponding CDT-face is marked (flagged) with a '1' and thus the DT-face is part of the boundary of the polyhedron. All other DT-faces are to be marked with a '0' to identify these faces as part of the interior or to be outside the polyhedron. The 'bitmap' of marked and non-marked lexical ordered faces (i.e. natural sorted on their nodes) is stored. To emphasize, the DT faces themselves are not stored, but only a bit for each face indicating whether or not the face is part of the boundary of the polyhedron. The CDT-faces marked with a '1', but not part of the DT, have to be stored separately.

In summery, the decomposition process consists of the following steps:

1. Create a CDT of the given polyhedron *P;*
2. Store CDT-nodes (hence nodes of polyhedron and extra Steiner points);

3. Mark CDT-faces (‘1’: face is part of boundary of polyhedron *P*, ‘0’ remaining faces);
4. Create DT of nodes CDT;
5. Mark DT-faces (‘1’: DT-face that corresponds to CDT ‘1 ‘ face is identical marked ‘1’; ‘0’ remaining faces);
6. Store ‘bitmap’ of marked and non-marked ordered DT-faces;
7. Store ‘1’-marked CDT-faces which do not have a corresponding DT-face.

The composition process consists of the following steps:

1. Read CDT-nodes;
2. Create DT of CDT-nodes;
3. Read ‘bitmap’
4. Order DT faces and set each DT-face according to the bitmap;
5. Create polygonal surface out of ‘1’-marked DT-faces;
6. Read non-recovered CDT-faces;
7. Create a watertight polyhedron boundary by completing the polygonal surface with non-recovered CDT-faces.

The gain in storage is within the difference of storing the polyhedron just by its nodes and the faces of its surface at the one hand, and the storage of the nodes of the CDT and the non-recovered CDT-faces at the other hand. To make this comparison more easy we assume the polyhedron is defined by a triangulated surface, thus all faces of the polyhedron are triangles. The storage requirements for this kind of triangular polyhedra are as follows: for each node of the polygon we need 3 floats for the X, Y and Z-coordinate, and for each face 3 integers (reference value to nodes).

The triangulated boundary of the polyhedron makes the comparison more easy, but is introduces one main disadvantage as is also stressed in section 1.2 of [25]: each triangular face of the boundary is now constrained, and thus more Steiner points are needed to make sure the CDT faces are part of the boundary of the polyhedron.

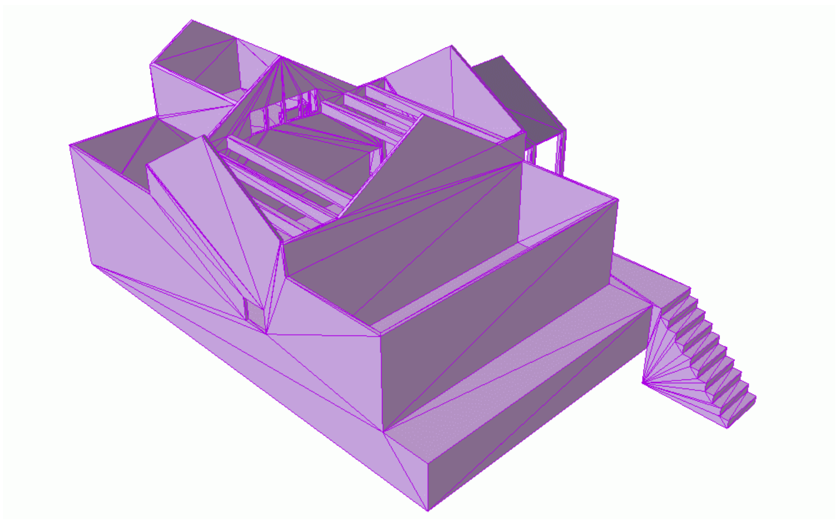


Fig. 7. Polyhedron representation dataset M440 with 906 surface faces

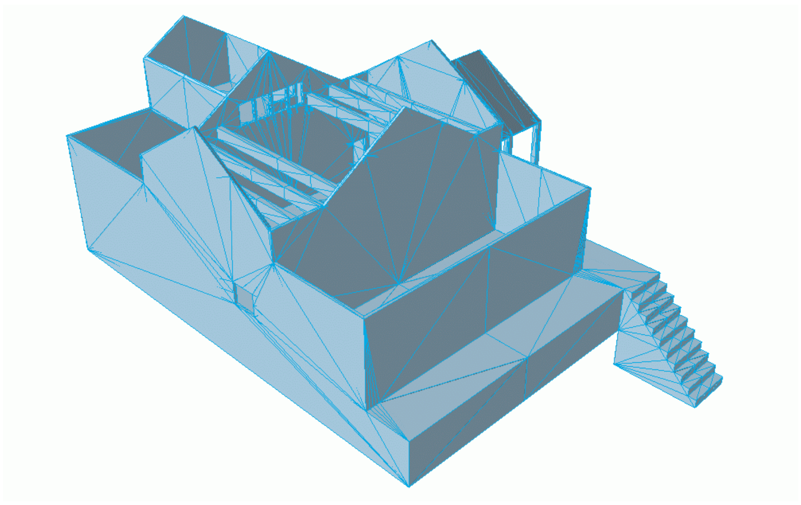


Fig. 8. Constrained Delaunay Tetrahedralization of Polyhedron representation of dataset M440

The storage requirements for the CDT encoded polyhedron sums to: for each node and also for all necessary Steiner points we need 3 floats for the X, Y and Z-coordinate, for each DT-face 1 bit, and for each non-recovered CDT-face 3 integers (reference values to nodes). As both methods require the storage of the nodes of the polygon, these are to be left out the comparison.

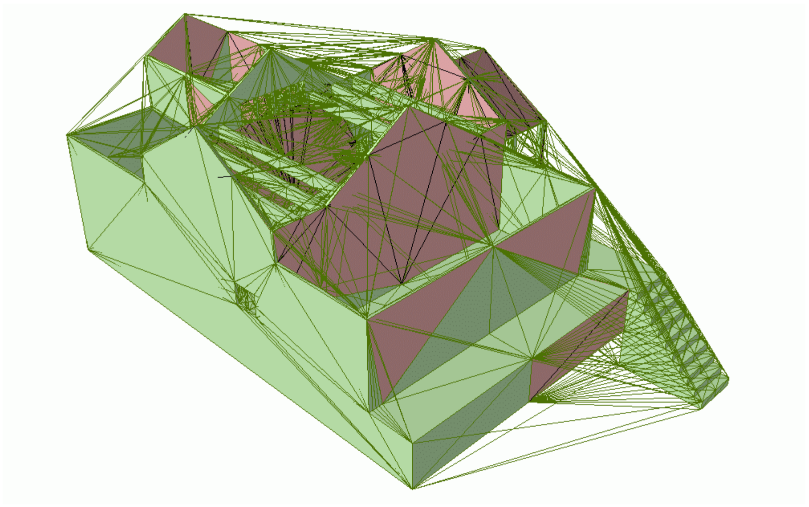


Fig. 9. Delaunay Tetrahedralization (DT) of vertices of Constrained Delaunay Tetrahedralization (CDT) of Polyhedron representation of dataset M440. The DT consists of 10545 faces. 187 Faces of the CDT are not recovered by the DT.

4.2 Experimental Results

We have tested this decomposition/composition method with some datasets from INRIAs 3D Meshed Research Database (`http://www-c.inria.fr/gamma/`). The tetrahedralization is performed by Tetgen 1.4.2 [31].

Polyhedron M440 represents a house, see Fig. 7. To store the 906 surface faces of the input model we need 906 * 3 * 32 = 86976 bits. The decomposition through means of a CDT as shown in Fig. 8, adds 480 Steiner Points, thus 480 * 3 * 32 = 46080 bits. The DT from the nodes of this CDT consists of 10545 faces (bits). The DT could not recover 187 faces of the CDT (see Fig. 9) so an extra 187 * 3 * 32 = 17952 bits are needed. In total we need: 46080 + 10545 + 17952 = 74577 bits. In conclusion, here we 'win' 86976 - 74577 = 12399 bits, or 14%.

For this dataset the gain is very modest. As mentioned in section 4.1, this result could be influenced by the triangulated boundary of the input polyhedron, as each face is to be recovered in the CDT.

5 Conclusions

We have demonstrated the utilization of the Constrained Delaunay Tetrahedralization (CDT) for the validation and efficient storage of polyhedra. The main advantage of this idea is not on the storage gain, but on the confidence to store closed, watertight polyhedral surfaces, and thus polyhedra. If a polyhedron is decomposable by a CDT, the surface faces of this CDT are to be connected to the surface of the polyhedron, and the interior of the CDT is connected, then the polyhedron is valid.

Further research could address the following questions:

- To what extent has a CDT to conform to the generic Delaunay Tetrahedralization? Adding more Steiner points can result in a more conforming or even completely Conforming Constrained Delaunay Tetrahedralization (CCDT). The efficiency is a trade off between this CCDT-likeness (with less non-CDT recovered faces to store) and the needed space to store the extra Steiner points and amount of faces of the Delaunay Tetrahedralization of the vertices of the CDT.
- What are the complications on validating and storing large datasets with many polyhedra collectively through one tetrahedralization? Is it still possible to validate a non-connected polyhedron or polyhedron completely inside another polyhedron? What will be the storage gain by the method as presented in this paper?

Acknowledgements

This research is carried out as part of the Dutch BSIK RGI program on 3D Topography (RGI-011). The authors like to express their gratefulness towards the reviewers of the GIScience 2008 conference and above all to Peter van Oosterom, Hugo Ledoux, Friso Penninga and Theo Tijssen for their rewarding comments.

References

[1] Arens, C., Stoter, J., van Oosterom, P.: Modelling 3D spatial objects in a geo-dbms using a 3D primitive. Computers & Geosciences 31, 165–177 (2005)

[2] OGC: Implementation specification for geographic information - simple feature access (2006), http://www.opengeospatial.org/standards/sfa

[3] Kazar, B.M., Kothuri, R., van Oosterom, P., Ravada, S.: On valid and invalid three-dimensional geometries. In: Advances in 3D Geoinformation Systems (2008)

[4] ISO: 19107 (2003), http://www.iso.org

[5] GML: The geographic markup language specification, version 3.3.1 (2003), http://www.opengeospatial.org/standards/gml

[6] van Oosterom, P., Quak, W., Tijssen, T.: About invalid, valid and clean polygons. In: Fisher, P.F. (ed.) Developments in Spatial Data Handling, 11th International Symposium on Spatial Data Handling, pp. 19–48 (2004)

[7] Kettner, L.: CGAL Manual. In: 3D Polyhedral Surfaces, ch. 12 (2008), http://www.cgal.org/Manual/3.3/doc_html/cgal_manual/Polyhedron/Chapter_main.html

[8] Verbree, E.: Encoding and decoding of planar maps through Conforming Delaunay Triangulations. In: ISPRS Workshop on multiple representation and interoperability of spatial data (2006)

[9] CGAL: Computational Geometry Algorithms Library (2007), http://www.cgal.org

[10] Verbree, E.: Piecewise linear complex representation through Conforming Delaunay Tetrahedronization. In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 385–388. Springer, Heidelberg (2006)

[11] Edelsbrunner, H., Tan, T.S.: An upper bound for conforming Delaunay triangulations. SIAM Journal on Computing 22, 527–551 (1993)

[12] Penninga, F., van Oosterom, P.: A Simplicial Complex-based DBMS Approach To 3D Topographic Data Modelling. International Journal of Geographical Information Science 22, 751–779 (2008)

[13] Penninga, F., van Oosterom, P.: Updating Features in a TEN-based DBMS approach for 3D Topographic Data Modelling. In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 147–152. Springer, Heidelberg (2006)

[14] Edelsbrunner, H.: Geometry and topology for mesh generation. Cambridge University Press, Cambridge (2001)

[15] Ziegler, G.M.: Lectures on Polytopes. Graduate Texts in Mathematics, vol. 152. Springer, New York (1995)

[16] Delaunay, B.N.: Sur la sphère vide. Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk 7, 793–800 (1934)

[17] Aurenhammer, F.: Voronoi diagrams – a study of fundamental geometric data structure. ACM Comput. Surveys 23, 345–405 (1991)

[18] Schönhardt, E.: Über die zerlegung von dreieckspolyedern in tetraeder. Mathematische Annalen 98, 309–312 (1928)

[19] Ruppert, J., Seidel, R.: On the difficulty of triangulating three-dimensional non-convex polyhedra. Discrete and Computational Geometry 7, 227–253 (1992)

[20] Murphy, M., Mount, D.M., Gable, C.W.: A point-placement strategy for conforming Delaunay tetrahedralizations. In: Proc. 11th annual ACM-SIAM Symposium on Discrete Algorithms, pp. 67–74 (2000)

[21] Cohen-Steiner, D., De Verdière, E.C., Yvinec, M.: Conforming Delaunay triangulation in 3D. In: Proc. 18th annual ACM Symposium on Computational Geometry (2002)
[22] Cheng, S.W., Poon, S.H.: Graded conforming Delaunay tetrahedralization with bounded radius-edge ratio. In: Proc. 14th annual ACM-SIAM Symposium on Discrete Algorithms, pp. 295–304 (2003)
[23] Chazelle, B.: Convex partition of a polyhedra: a lower bound and worest-case optimal algorithm. SIAM Journal on Computing 13, 488–507 (1984)
[24] Chazelle, B., Palios, L.: Triangulating a nonconvex polytope. Discrete and Computational Geometry 5, 505–526 (1990)
[25] Shewchuk, J.R.: General-dimensional constrained Delaunay and constrained regular triangulations I: Combinatorial properties. Discrete and Computational Geometry (2008)
[26] Si, H., Gärtner, K.: Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. In: Proc. 14th International Meshing Roundtable, San Diego, CA, USA, Sandia National Laboratories, pp. 147–163 (2005)
[27] Bern, M.: Compatible tetrahedralizations. In: Proc. 9th annual ACM Symposium on Computational Geometry, pp. 281–288 (1993)
[28] Liu, A., Baida, M.: How far flipping can go towards 3D conforming/constrained triangulation. In: Proc. 9th International Meshing Roundtable, Sandia National Laboratories, pp. 307–315 (2000)
[29] Karamete, B.K., Beall, M.W., Shephard, M.S.: Triangulation of arbitrary polyhedra to support automatic mesh generators. International Journal for Numerical Methods in Engineering 49, 167–191 (2000)
[30] George, P.L., Borouchaki, H., Saltel, E.: Ultimate robustness in meshing an arbitrary polyhedron. International Journal for Numerical Methods in Engineering 58, 1061–1089 (2003)
[31] Si, H.: TetGen (2007), `http://tetgen.berlios.de`
[32] Möller, T.: A fast triangle-triangle intersection test. Journal of Graphics Tools 2, 25–30 (1997)
[33] Guigue, P., Devillers, O.: Fast and robust triangle-triangle overlap test using orientation predicates. Journal of graphics tools 8, 39–52 (2003)
[34] Hoffmann, C.M.: Geometric and Solid Modeling – An Introduction. Morgan Kaufmann, San Mateo, CA (1989), `http://www.cs.purdue.edu/homes/cmh/distribution/books/geo.html`
[35] Dyken, C., Floater, M.S.: Preferred directions for resolving the non-uniqueness of Delaunay triangulations. Computational Geometry: Theory and Applications 34, 96–101 (2006)

Ontology-Based Geospatial Data Query and Integration

Tian Zhao[1], Chuanrong Zhang[2], Mingzhen Wei[3], and Zhong-Ren Peng[4]

[1] Dept. of Computer Science, Univ. of Wisconsin – Milwaukee, Milwaukee, WI
[2] Dept. of Geography & the Center for Environmental Sciences and Engineering, Univ. of Connecticut, Storrs, CT
[3] U.S. Geological Survey, Rolla, MO
[4] Dept. of Urban and Regional Planning, Univ. of Florida, Gainesville, FL, College of Transportation Engineering, Tongji University, Shanghai, China

Abstract. Geospatial data sharing is an increasingly important subject as large amount of data is produced by a variety of sources, stored in incompatible formats, and accessible through different GIS applications. Past efforts to enable sharing have produced standardized data format such as GML and data access protocols such as Web Feature Service (WFS). While these standards help enabling client applications to gain access to heterogeneous data stored in different formats from diverse sources, the usability of the access is limited due to the lack of data semantics encoded in the WFS feature types. Past research has used ontology languages to describe the semantics of geospatial data but ontology-based queries cannot be applied directly to legacy data stored in databases or shapefiles, or to feature data in WFS services. This paper presents a method to enable ontology query on spatial data available from WFS services and on data stored in databases. We do not create ontology instances explicitly and thus avoid the problems of data replication. Instead, user queries are rewritten to WFS getFeature requests and SQL queries to database. The method also has the benefits of being able to utilize existing tools of databases, WFS, and GML while enabling query based on ontology semantics.

1 Introduction

The National Spatial Data Infrastructure promotes geospatial data sharing to improve data quality, to reduce costs, and to make data more usable to the public [17]. However, much of the existing geospatial data is stored in proprietary formats such as ESRI shapefiles, coverage, and geodatabases, and it is only accessible through vendor-specific geospatial information systems such as ESRI ArcGIS and Intergraph GeoMedia [29,36]. Data sharing is difficult in this context because different GIS software has incompatible system designs, data models, and database storage structures [9,28,34,35].

To enable sharing of geospatial data, the Open Geospatial Consortium (OGC) has established a series of specifications such as Geographic Markup Language (GML) [11] and Filter encoding, and data exchange protocols such as Web Feature Service (WFS) [13] and Web Map Service (WMS) [12]. These specifications and protocols have provided constructs for describing and accessing geospatial data at the domain level. For example, GML files can encode the geometries of an object as points, lines, or polygons. The WFS server can accept query of features within a bounding box and return the query results in GML format. The resulting GML data can be rendered for visualization based on the geometry encoding. Also, WFS servers accept more complex queries using OGC filter encodings with relational or spatial operators [13]. Although the fast development of these standards and web service technologies has undoubtedly improved the sharing and synchronization of geospatial information across diverse resources, they only can support technical data interoperability and cannot resolve semantic heterogeneity problems in spatial data sharing. WFS and WMS protocols, and GML and OGC filters have no provisions for data sharing at semantics level for applications. For example, a WFS server may name a feature representing a bus route as “Route” or “ROUTE”. A *getFeature* query to the WFS server has to spell the name correctly, otherwise no results will return. Similarly, the geometry of the feature can be either a complete route or just a link segment of the route. WFS service client has to know this information in order to formulate *getFeature* requests to retrieve a route by its name. Moreover, a route may have implicit relations with other features such as bus stop but such relation is not specified as a feature property. Even if relations such as bus stops of a route are somehow included as feature properties, they has to be encoded using XML complex types because of the one-to-many relations and users of WFS services may not be able to interpret the meaning of these feature properties based on the XML types alone.

The root of the problem is that structured data such as XML and GML does not have enough constructs to express data semantics. As a result, software developed based on the OGC standards cannot be readily adapted to consider data semantics. Recent research [18,1,15,28,36] has applied the concepts of semantic web to geospatial data sharing. Semantic web [5] promotes the use of ontology languages such as Resource Description Framework (RDF) and Web Ontology Language (OWL) to provide semantics for data usually found in databases or other structured documents [19,36]. One important advantage of RDF and OWL ontology is that it is easier to define semantics for data using ontology constructs such as classes and properties. In addition, combined with ontology semantic constraints, the ontology can allow reasoners based on Description Logic (DL) to infer further knowledge from partially specified data and check for data consistencies [2]. Tools like Jena[1] allow inference rules be used to support more powerful query of ontology data. For example, in transit system, a property *transitStop* may be the same as the composition of the *transitPointFeatureEvent* and *stopEvent* properties. This is not yet supported in OWL but one can encode this rule in Jena using its general purpose rule engine. Finally, ontology definitions are extensible so that geospatial applications can use existing domain ontologies as basis to create application ontologies. Thus, by providing a semantic interpretation of the data, RDF and OWL ontology allows software programs to automatically understand structures and meanings of diverse information sources and conduct automatic knowledge inference or reasoning from existing data and documents.

Recent research projects have applied ontology to modeling observations and measurements [30], to enable spatial/temporal/thematic reasoning [1], and to assist the discovery and access of geospatial web services [27]. However, ontology reasoners only can apply to ontology instances such as RDF instances or OWL individuals, which correspond to the database records and WFS feature instances. Transforming all legacy

[1] http://jena.sourceforge.net/

geospatial data to ontology form is time consuming, error prone, and inefficient. Also ontology tools such as Protégé[2] cannot efficiently manipulate ontology instances in large quantity due to memory consumption. Moreover, it is not cost-effective for ontology tools to include the functionalities provided by geodatabases and WFS services such as transaction management and spatial query. Thus, to efficiently support ontology-based reasoning on geospatial data, it is necessary to keep legacy data stored in geodatabases and other data files while providing an ontology-enabled interface to translate user requests into queries to legacy data stores. To this end, there are projects focused on extracting ontology definitions from database schemas [3] and annotating geospatial data with semantic annotations based on OWL and inference rules [23]. However, it is not clear how the extracted ontology information can help translate user requests into queries to legacy data sources.

In this paper, we propose a new solution to spatial data interoperability at semantic level through an interface based on RDF ontology. We use a real world transportation application, a transit system, as an example for our solution. Comparing with other existing approaches, the most important advantage of our solution is that it does not need to replicate legacy data stored in relational databases, shapefiles, or GML data accessible from WFS services. The interface is to provide an ontology layer for spatial data accessible from WFS services and databases. Users of the interface can query spatial data as if it is defined in terms of some domain and application ontologies. The interface relies on WFS services as data sources for spatial data so that it can use the spatial query functions of WFS servers and use existing WFS client library for feature rendering. The interface can also use relational databases as sources for non-spatial data since this can improve the performance of queries not involving geometries.

The rest of the paper is organized as follows. We discuss related works in Section 2. In Section 3, we give an overview of the proposed interface. Section 4 presents a simple RDF ontology used in our example. In Section 5, we describe the mapping and inference rules that are used to connect ontology definitions with the feature data in databases and WFS servers. Section 6 explains the supported RDF queries, the query semantics, our query rewriting algorithm, and possible extension to handle RDF semantic constraints. Implementation issues are discussed in Section 7.

2 Related Work

RDF-based data integration The problem of data integration is to combine data of different sources and provide users a unified view of the data [24]. A data integration system often uses a global schema containing mappings from global definitions to local schemas in each data sources. In this context, data query problem can be reduced to the problem of answering queries using materialized views [21], where data sources are described as precomputed views on the global schema. Our method is similar to answering queries using views except that the global schema is defined using RDF ontology and because of this, the views are not just the mapping from data sources on the global schema but also include inference rules to obtain object properties. Also, we need to consider the semantic constraints of the RDF ontology such as subclass

[2] http://protege.stanford.edu

and subproperty relations. In this aspect, our method is closely related to [8] that also uses RDF ontology as a medium to provide integrated access to different relational databases. Their approach is to define database schemas as views on RDF ontology entirely including object properties and semantic constraints. One of their focus is to consider databases as incomplete data sources such that missing information is tolerated while query rewriting may result in alternative answers to the same query based on different data sources used. In comparison, our method is more restrictive in the way that mappings may be defined from database tables or WFS features to RDF ontology. In particular, we require each database table and WFS feature to map to one RDF class. Additional RDF classes and object properties are defined through inference rules. This simplifies the query rewriting algorithm and still preserves the flexibility of using RDF ontology to express complex relations of spatial and non-spatial objects. Another related work is D2RQ [7], which is a tool suite that provides RDF interface to relational databases. D2RQ allows users to define a mapping file with rules to establish a RDF ontology that maps RDF classes and properties to database tables and columns. Also, additional properties may be defined over existing RDF classes and properties in a way similar to our inference rules. D2RQ only provides RDF interface for relational databases. We use it to provide access to non-spatial data in databases while spatial data queries are handled by WFS servers.

Geo-spatial ontology. Difference in semantics used in different data sources is one of the major problems in spatial data sharing and data interoperability [6,16]. One possible approach to overcome the problem of semantic heterogeneity is by means of ontology [18,31,32]. Cruz *et. al.* [10] studied the problem of ontology-alignment in the context of integrating geospatial data in databases. They developed a semi-automatic method of generating mappings between ontologies of local databases and a global one. The mappings can then be used for query rewriting. Baglioni *et. al* [3] proposed a method on accessing spatial database through ontology layer by semi-automatically building an application ontology from a geographical database and then enrich it with domain ontology by finding correspondence between the classes and properties of the two ontologies. The enriched ontology is said to be used in assisting query answering though it is not clear how semantic queries are translated to spatial SQL to databases. Also related is a project for semi-automatically adding semantic annotations to geospatial data [23], which uses OWL to provide semantic annotation and uses Semantic Web Rule Language (SWRL)[3] to add additional properties between instances based on the existing ontology. More general discussion on developing geospatial ontologies can be found in the work of Arpinar *et. al.* [1], where geospatial semantics are considered for three types of geospatial relations: topological relations, cardinal directions, and proximity relations. Their system architecture would pull geospatial data from sources such as National Map, NASA sources, UCGIS sources, and put it in a massive metadata store, which is then used to populate the ontology-based knowledge base accessible to users via spatial/temporal/thematic reasoners. When geospatial data is solely provided through web services, an additional layer of ontology included in a Service Oriented Architecture can be useful. Paul and Ghosh [27] argue that a domain ontology can be

[3] http://www.w3.org/Submission/SWRL/

used to provide shared vocabulary for the schemas of WFS servers. User queries to a service broker, which maintains a list of services, can somehow be translated to specific getFeature requests to different service providers – WFS servers. The SPIRIT spatial search engine [22] has shown ontology to be useful in searching web documents with spatial contents. User queries can include a subject, a place name, and a spatial relation to the place name. Results are list of documents and their positions on a map. The search engine uses geographical and domain ontologies to disambiguate and expand user queries, to rank documents based on relevance, and to extract metadata from web documents.

The overall assessment is that our approach may extend or complement the related works. For example, some methods [3,10] generate useful ontologies from geospatial databases, which can be used as the application and/or domain ontologies for our interface. We do not use OWL as in [23] but it may be possible to include OWL ontologies with help of reasoning tools. Also, our method could be one step towards realizing the goal of using reasoners to query data sources through ontology interface as proposed in [1]. The differences are that we do not require data be migrated from sources to knowledge base, thus avoiding the problems of data replication, and we do not support thematic and spatial-temporal queries. Lastly, our method may be applied to both web service discovery and retrieval [27]. In this setting, web services should be published using the class and properties of a domain ontology and service brokers then translate service-retrieval requests to WFS getFeature requests using predefined rules.

3 A RDF-Based Spatial Data Interface

Geospatial data contains both spatial attributes such as geometry and non-spatial attributes. The fact that spatial data is distributed among many sources and stored in different formats makes it hard to query spatial data through a single interface. WFS as a standardized protocol designed to alleviate this problem by providing uniform interface to data stores in forms of databases, shapefiles, and GML files. Though there are sophisticated WFS server implementations such as GeoServer[4], software for implementing WFS clients is less than ideal. The client software such as MapBuilder[5] and OpenLayers[6] primarily supports map retrieval, feature rendering, and feature transactions. While the existing WFS clients can locate WFS services and create map layers, the utility of these clients is limited by the lack of data semantics in WFS features. Also, though features may contain descriptive attributes, joint queries based on them are not very efficient since the WFS protocol uses bulky GML tags to encode feature attributes while WFS service is more suitable for spatial querying and transactions. Many identifying attributes of the spatial features can be efficiently stored and accessed through traditional relational databases.

In this paper, we describe an ontology-based interface for querying spatial features. We focus on three aspects:

[4] http://geoserver.org/display/GEOS/GeoServer+Home
[5] http://communitymapbuilder.org/
[6] http://www.openlayers.org/

```
<rdfs:Class rdf:ID="Feature"/>

<rdfs:Class rdf:ID="SpatialFeature">
  <rdfs:subClassOf rdf:resource="#Feature"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Geometry"/>

<rdf:Property rdf:ID="hasID">
  <rdfs:domain rdf:resource="#Feature"/>
</rdf:Property>

<rdf:Property rdf:ID="geometry">
  <rdfs:domain rdf:resource="#SpatialFeature"/>
  <rdfs:range rdf:resource="#Geometry"/>
</rdf:Property>
```

Fig. 1. Domain ontology in RDF schema

1. Create an ontology to describe the problem domain of the data supported by the interface.
2. Define mapping and inference rules to connect the ontology with the WFS feature types and database schemas.
3. Rewrite ontology queries to getFeature requests to WFS services and SQL queries to databases to obtain answers.

For simplicity, we assume that the spatial data in question is distributed among a WFS service and a relational database. The WFS service stores the feature geometries and IDs while the relational database stores the rest of the non-spatial attributes.

We define the ontology in terms of a RDF schema. In the schema, RDF classes correspond to the feature types while RDF properties correspond to feature attributes and relations between features. The instances of the RDF classes form a RDF graph and they correspond to the spatial features but we are not going to create these RDF instances explicitly. Instead, the ontology interface provides access to the features through query rewriting. The mapping rules define the correspondence between the RDF properties and the WFS feature attributes and database columns. Simply mapping rules are not very useful other than unifying the access to the two data sources. We use additional inference rules to define some RDF properties between RDF instances in terms of other RDF properties created in the previous step.

4 RDF-Based Ontology

We begin with a simple domain ontology where `Feature` class contains all spatial and non-spatial objects. All instances of this class have an ID field through the property `hasID`. Spatial objects belong to the `SpatialFeature` class, which is a subclass of `Feature` and has an additional `geometry` property.

Class	Superclass
Route	*Feature*
Pattern	*Feature*
LinkSequence	*Feature*
Stop	*SpatialFeature*
Link	*SpatialFeature*

Property	Domain	Range
name	*Route*	*string*
intersection	*Stop*	string
routeID	*Pattern*	integer
patternID	*LinkSequence*	integer
linkID	*LinkSequence*	integer
linkOrder	*LinkSequence*	integer
route_link	*Route*	*Link*
route_stop	*Route*	*Stop*
nearby_stop	*Stop*	*Stop*

Fig. 2. Summary of application ontology for transit system

We extend the domain ontology with an application ontology to describe a transit system with `Route`, `Stop`, `Link`, `LinkSequence`, `Pattern`, etc. The `Stop` and `Link` are spatial classes with point and line geometry respectively. Other classes describe non-spatial features but they are all closely related to the spatial features. For example, each instance of `Route` class contains several instances of `Link` that make up the route through the property `route_link`. A `Route` instance can also point to several instances of `Stop` that are on the route. A `Stop` instance can refer to stops within a certain radius via the property `nearby_stop`. We can find the `intersection` of a `Stop` instance as well.

Other classes and properties are not directly related to spatial objects but they are used to infer the properties such as `route_link`. For example, the spatial data for link and stop originally stored as shapefiles may be accessible as Web Features through a WFS server. However, they only contain IDs and their geometries. Thus, we cannot find out the property `route_link` directly by querying the WFS service, and we need other classes and properties to infer `route_link`. The non-spatial data such as Route, Stop (some of its properties), LinkSequence, Pattern often is stored in tables of a relational database and we can use its properties to infer `route_link`. Specifically, we can use the Pattern table to find the patterns in a route and then find the links contained in these patterns through the LinkSequence table (which describes the links traversed in a pattern sequentially).

5 RDF Views of WFS Features and Database Tables

The RDF ontology is an abstraction of the spatial and non-spatial data stored in WFS servers and relation databases. In order to have an ontology-based interface to the data, we need a way to map the RDF ontology to the schemas of WFS features and relational tables. This mapping is defined as *RDF views* from the feature and database schemas to the ontology. With RDF views, RDF queries can be rewritten to *getFeature* requests to WFS servers and SQL queries to databases.

The RDF views are defined in two steps. The first step is to create mapping rules to define each WFS feature and relational table as a view over the RDF ontology. In this step, only *datatype properties* are used in the mapping rules. Datatype properties are the properties with ranges of primitive types such as string or integer. The second step is to define inference rules for some additional RDF properties in terms of the datatype properties used in the previous step. The additional properties include the so-called *object properties* whose ranges are RDF class types.

5.1 Mapping Rules

Definition 1. *A mapping rule from WFS feature or database table to RDF triples has the form of* $p(\overline{X}) : -R(\overline{X}, \overline{Y})$*, where* p *is a predicate corresponding to a WFS feature or a relational table,* R *is a set of RDF triples,* $\overline{X}$ *and* $\overline{Y}$ *are sets of variables.*

Figure 3 illustrates the mapping from relational tables and WFS features to RDF ontology. The mapping rules written in Datalog-like[7] notations are summarized in Figure 4. A mapping rule m has two parts, the left of `:-` is called the rule head and can be written as $m.head$ and the right of `:-` is the rule body accessible via $m.body$. The rule head is a predicate corresponding to a WFS feature or database table. For example, `wfs:link(?id, ?geom)` refers to the link feature in a WFS server, where $?id$ and $?geom$ are variables corresponding to the ID and Geometry properties respectively. Note that any string started with ? is a variable. Similarly, `db:Route(?id, ?name)` is a route table in a database where $?id$ and $?name$ are variables representing the ID and name columns of the route.

The rule body is a set of RDF triples written in N3[8] notations, where each triple has the form of `subject predicate object`, and $subject$ and $object$ can be variables that correspond to RDF instances or primitive values such as strings, and $object$ can also be RDF types or constants. The predicate corresponds to the RDF properties such as `hasID` and `geometry`. For each database table and each WFS feature type, we create a corresponding RDF class. The mapping rules map the database tables and WFS feature types to the corresponding RDF classes. Consequently, the RDF triples in a body of a rule always have the same subject. For example, the only subject in Rule (M1) is `?stop` while the subject in (M2) is `?link`.

In our definition, a variable $?x$ appearing in the rule head corresponds to the value of a WFS feature property or a database table cell, while a variable $?y$ only appearing in the rule body but not in rule head corresponds to RDF instances. Since we don't create RDF instances explicitly, the variables such as $?y$ are never materialized.

5.2 Inference Rules

The mapping rules connect WFS features and database tables to RDF ontology but only the datatype properties of the ontology are used. RDF ontology is more flexible in that it can define object properties to connect RDF instances. For example, we can define `route_link` property to specify the links contained in a route. This is useful because

[7] http://www.ccs.neu.edu/home/ramsdell/tools/datalog/datalog.html
[8] http://www.w3.org/2000/10/swap/Primer.html

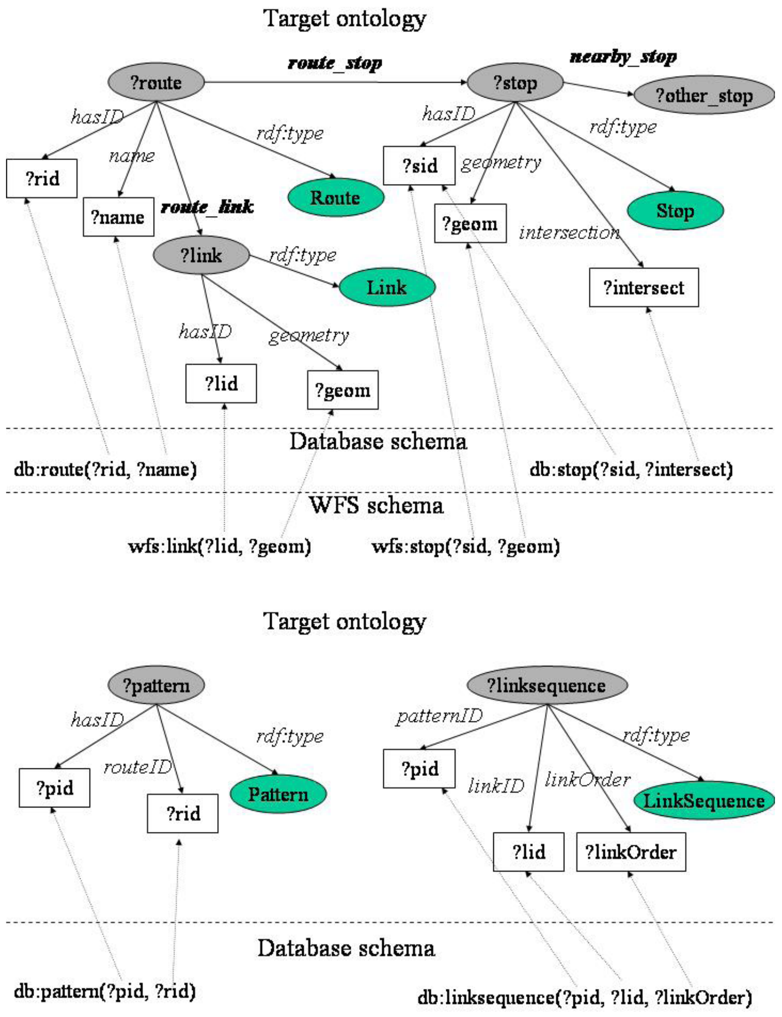


Fig. 3. Schematic mapping from database tables and WFS features to RDF ontology

a route does not have geometry of its own. If we want to display a route on a map based on the route name, we cannot rely on the datatype properties of the `Route` class. Instead, we can use `route_link` property to find out the links in a route with certain name and render the geometries of these links on a vector layer of a map.

To connect object properties such as `route_link` to WFS features and database tables, we use a set of inference rules to derive object properties based on the datatype properties. An inference rule i also written in Datalog-like notations has two parts: the head written as $i.head$ and the body written as $i.body$. The head of the rule has the form of a RDF triple while the body is a set of RDF triples plus some filters.

```
(M1) wfs:stop(?geom, ?sid)  :- ?stop rdf:type Stop,
                               ?stop hasID ?sid,
                               ?stop geometry ?geom.
(M2) wfs:link(?geom, ?lid)  :- ?link rdf:type Link,
                               ?link hasID ?lid,
                               ?link geometry ?geom.
(M3) db:stop(?sid, ?intersect)
                            :- ?stop rdf:type Stop,
                               ?stop hasID ?sid,
                               ?stop intersection ?intersect.
(M4) db:route(?rid, ?name)  :- ?route rdf:type Route,
                               ?route hasID ?rid,
                               ?route name ?name.
(M5) db:pattern(?pid, ?rid) :- ?pattern rdf:type Pattern,
                               ?pattern hasID ?pid,
                               ?pattern routeID ?rid.
(M6) db:linksequence(?pid, ?lid, ?linkOrder)
                            :- ?linkseq rdf:type LinkSequence,
                               ?linkseq patternID ?pid,
                               ?linkseq linkID ?lid,
                               ?linkseq linkOrder ?linkOrder.
```

Fig. 4. Mapping rules from WFS features and relational tables to RDF ontology

```
(I1)                                 (I2)
?route route_link ?link :-
   ?route rdf:type Route,
   ?route hasID ?rid,
   ?pattern rdf:type Pattern,        ?stop nearby_stop ?other :-
   ?pattern hasID ?pid,                 ?stop rdf:type Stop,
   ?pattern routeID ?rid,               ?other rdf:type Stop,
   ?linkseq rdf:type LinkSequence,      ?stop geometry ?geom,
   ?linkseq hasID ?lid,                 ?other geometry ?g,
   ?linkseq patternID ?pid,             filter(
   ?link rdf:type Link                     DWithin(?g, ?geom, 1)
   ?link hasID ?lid.                    ).
```

Fig. 5. Inference rules where `filter(DWithin(?g, ?geom, 1))` is a filter to specify that ?g is within a distance of 1 from ?geom

Definition 2. *An inference rule has the form of* $r(\overline{X}) : -R(\overline{Y}, \overline{Z}), F(\overline{Z})$*, where* r *is a RDF triple,* R *is a set of RDF triples,* F *is an optional set of filters, and* $\overline{X}$*,* $\overline{Y}$*,* $\overline{Z}$ *are sets of variables.* $\overline{X}$ *is a subset of* $\overline{Y}$ *and they contain variables referring to RDF instances.* $\overline{Z}$ *is a set of variables referring to datatype values or geometries.*

Figure 5 shows two inference rules where Rule I1 says that Property `route_stop` relates a route to a link if the route ID matches the route ID of a pattern, the pattern's ID matches the pattern ID of a link-sequence, and the link-sequence's link ID matches the link's ID. The inference rule for `route_stop` can be defined similarly. Note that the filter in Rule I2 is similar to the OGC filters and the rule says that a stop one has a nearby stop two if stop two's geometry is within a distance of 1 from stop one's geometry. Here we omit the unit of distance.

To simplify query rewriting algorithm, we require an object property to appear in the head of at most one inference rule.

6 RDF Queries

We use SPARQL[9] – a query language for RDF data to write our spatial queries. The SPARQL queries have to be rewritten to WFS getFeature requests and SQL queries. The queried results can then be displayed or rendered on a map.

Definition 3. *We consider SPARQL queries in the form of*

$$select\ \overline{X}\ where\ R(\overline{Y}, \overline{Z}),\ F(\overline{Z})$$

where R is a set of RDF triples, F is an optional set of filters, and $\overline{X}$, $\overline{Y}$, and $\overline{Z}$ are set of variables. $\overline{X}$ is a subset of $\overline{Z}$. $\overline{Y}$ is a set of variables corresponding to RDF instances while $\overline{Z}$ is a set of variables corresponding to datatype values and geometries.

We restrict the set of variables that a SPARQL query selects to be datatype variables or geometries. The reason is that we are mainly interested in the attributes of spatial or non-spatial features, not the virtual RDF instances. We call the set of RDF triples in a SPARQL query q its body and write it as $q.body$.

As an example, suppose we want to find the geometry of a route by the route name "Summit". The SPARQL query can be written as

```
(Q1) select ?geom where ?route rdf:type Route.
                        ?route route_link ?link.
                        ?link geometry ?geom.
                        ?route name ?name.
                        filter(?name = "Summit").
```

Here we use the property `route_link` to find the links of a route with name "Summit". Another example is to find the intersection address of a bus stop based on the x, y coordinates of the stop (-88, 43).

```
(Q2)
select ?intersect where ?stop rdf:type Stop.
                        ?stop intersection ?intersect.
                        ?stop geometry ?geom.
                        filter(DWithin(?geom, (-88,43), 0.1)).
```

[9] http://www.w3.org/TR/rdf-sparql-query/

To display the nearby stops of a given stop, we can use the query below:

```
(Q3) select ?g where ?stop rdf:type Stop.
                     ?stop geometry ?geom.
                     filter(DWithin(?geom, (-88,43), 0.1)).
                     ?stop nearby_stop ?other.
                     ?other geometry ?g.
```

Note that a user interface can hard-wire some of the queries into the interface so that users do not have to write them explicitly. The queries such as Q2 and Q3 are inconvenient for users to write directly because the geometries of interests are difficult to specify precisely. The interface, however, can rely on mouse click to select stops and then query for the nearby stops or intersection addresses.

6.1 Query Semantics

Given a set of RDF views connecting a RDF ontology and a set of WFS and database schemas, we can always convert a set of WFS and database instances – the source instances, to a set of RDF instances – the target instances. The goal here is to allow querying on the target instances without actually creating them.

Thus, the problem we want to solve is

> to find the answers to a SPARQL query Q1 on the RDF ontology by rewriting the query to WFS/SQL queries Q2 so that the answer to Q2 on source instances is the same as the answer to Q1 on target instances.

WFS feature instances

```
   wfs:stop(10, (-88,42))          wfs:link(22, ((-88,42), (-88,43)))
   wfs:stop(11, (-88,43))          wfs:link(23, ((-88,43), (-87,43)))
```

database instances

```
       db:route(25, "Summit")          db:pattern(11, 25)
       db:stop(10, "Main at Oakland")  db:linksequence(11, 22, 1)
       db:stop(11, "Main at Adams")    db:linksequence(11, 23, 2)
```

target RDF instances

```
_stop10 rdf:type     Stop;                 _route25 rdf:type Route;
        hasID        10;                            hasID    25;
        intersection "Main at Oakland";             name     "Summit".
        geometry      (-88,42).
                                           _pattern11 rdf:type Pattern;
_stop11 rdf:type     Stop;                              hasID    11;
        hasID        11;                                routeID  25.
        intersection "Main at Adams";
        geometry      (-88,43)             _seq11_1  rdf:type  LinkSequence;
                                                     linkID    22;
_link22 rdf:type Link;                               linkOrder 1;
        hasID    22;                                 patternID 11.
        geometry ((-88,42), (-88,43)).
                                           _seq11_2  rdf:type  LinkSequence;
_link23 rdf:type Link;                               linkID    23;
        hasID    23;                                 linkOrder 1;
        geometry ((-88,43), (-87,43)).               patternID 11.
```

Fig. 6. WFS/database instances and the corresponding RDF instances written in N3 notations. For example, _route25 is a node of Route type and it has ID 25 and name "Summit"

```
_route25 rdf:type     Route;    _stop10 rdf:type      Stop;
         hasID        25;               hasID         10;
         name         "Summit";         intersection  "Main at Oakland";
         route_link   _link22;          geometry      (-88,42);
         route_link   _link23;          nearby_stop   _stop11.
                                _stop11 rdf:type      Stop;
                                        hasID         11;
                                        intersection  "Main at Adams";
                                        geometry      (-88,43);
                                        nearby_stop   _stop10.
```

Fig. 7. RDF node with properties derived with inference rules

As an example, consider the WFS and database instances shown in Figure 6, where `wfs:stop(10, (-88,42))` represents a feature instance of bus stop that has ID 10 and point geometry $(-88,42)$. Similarly, the line geometry of `wfs:link(22, ((-88,42),(-88,43)))` is $((-88,42),(-88,43)))$. Also, a database instance written as `db:linksequence(11, 22, 1)` is a link sequence with pattern ID 11, link ID is 22, and link order 1.

We use the mapping rules to create target RDF instances shown in Figure 6. Applying the inference rule I1 and I2 to the sample data, we obtain additional properties for the instances of Route and Stop class as shown in Figure 7. Figure 8 illustrates the relationship between the route nodes, link nodes, and stop nodes. If we directly apply query Q1 to the target instances in Figure 6 and 7, then we can find that a route by the name of "Summit" has two links `_link22` and `_link23`, which have the line geometries of `((-88,42), (-88,43))` and `((-88,43), (-87,43))`. We want to rewrite Q1 to WFS requests and SQL queries so that the same answers are returned from the WFS servers and databases.

6.2 Query Rewriting

The query rewriting algorithms have two parts. The first part shown in Figure 9 applies inference rules to the body of a SPARQL query (*target query*) so that RDF triples with object properties are replaced by RDF triples with datatype properties. An inference

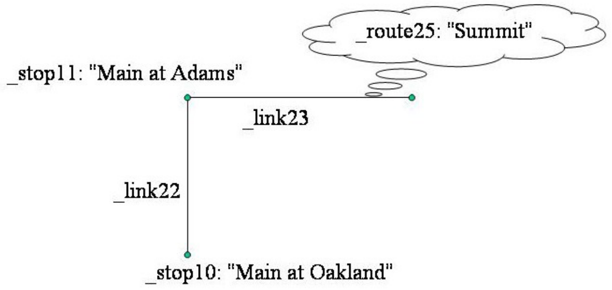


Fig. 8. Route _route25 includes the links _link22, _link23 and the stops _stop10 and _stop11

```
input: target query q
       a set of inference rules I

for each triple t in q.body
  for each inference rule i in I
     if there exists a substitution s such that s(i.head) = t
     then replace t in q.body with s(i.body)
  end for
end for

output: q' where q'.body has only triples in RDF mapping
```

Fig. 9. Algorithm 1: apply inference rule to SPARQL query

rule i is applicable to a triple t if $i.head$ matches t via a variable substitution s such that $s(i.head) = t$.

Definition 4. *A substitution s is a variable mapping from the set V to another set V' such that for each $v \in V$, $s(v) \in V'$. $s(t)$ is t with every variable in t replaced by $t(v)$.*

Note that an inference rule may define an object property in terms of not only datatype properties but also other object properties. This problem can be solved by recursively applying the inference rules to the query body until no object property remains. We do not consider this case for simplicity

Also note that the inference step is similar to *backward chaining* technique of automatic deduction used in logic programming systems such as Prolog [33]. However, we do not define recursive inference rules and the head of each inference rule is distinct. The restrictions ensure that Algorithm 1 always terminates and there is no need for backtracking (to try alternative rules of the same head).

The second part shown in Figure 10 applies algorithm 1 to the target query, and then rewrites the resulting query to WFS *getFeature* requests and SQL queries.

For example, we can apply inference rule I1 to the query Q1 to replace `?route route_link ?link.` with the body of Rule I1. Below is the result with some redundant triples removed.

```
(Q1') select ?geom where ?route rdf:type Route.
                         ?route hasID ?rid.
                         ?route name ?name.
                         filter(?name = "Summit").
                         ?pattern rdf:type Pattern.
                         ?pattern hasID ?pid.
                         ?pattern routeID ?rid.
                         ?linkseq rdf:type LinkSequence.
                         ?linkseq patternID ?pid.
                         ?linkseq hasID ?lid.
                         ?link rdf:type Link.
                         ?link hasID ?lid.
                         ?link geometry ?geom.
```

```
input: target query q
       a set of mapping rules M
initialize: apply algorithm 1 to q to obtain q′
            group triples in q′.body by subject name
            resulting a set of triple groups L

for each triple group t in L
   for each mapping rule m in M
      if there exists a substitution s such that
              s(m.body) contains all triples in t
      then replace t in q′ with s(m.head)
   end for
end for

output: q′ where q′.body contains database queries and/or
        WFS getFeature requests
```

Fig. 10. Algorithm 2: query rewriting

Next, we apply mapping rules to replace the query body with WFS requests and SQL queries. For Query `Q1′`, we apply the set of mapping rules M2, M4, M5, and M6. After substitution, we get query Q1”.

```
(Q1′′) select ?geom where wfs:link(?geom, ?lid),
                          db:pattern(?pid, ?rid),
                          db:linksequence(?pid, ?lid, ?linkOrder),
                          db:route(?rid, ?name),
                          filter(?name = "Summit").
```

At this point, the query rewriting is complete where the body of Query `Q1′′` has a WFS *getFeature* request for the link feature and and a database SQL query for the pattern, linksequence, and route tables.

The order of execution of this query may be significant, however, since it is not efficient to send the WFS *getFeature* request without obtaining link IDs of the route “Summit” first. There could be large number of link features and the resulting GML file returned by the *getFeature* request can be slow to download and parse. This requires us to execute the database query that corresponds to:

```
db:pattern(?pid, ?rid),
db:linksequence(?pid, ?lid, ?linkOrder),
db:route(?rid, ?name),
filter(?name = "Summit").
```

This fragment can be translated to SQL of the form

```
select distinct l.lid
from patterns p, linksequence l, route r
where p.pid = l.pid and
      r.rid = p.rid and
      r.name = "Summit";
```

The retrieved link IDs are used to send *getFeature* request to the WFS server for the geometries of the feature `link`.

Similarly, Query Q2 can be rewritten as

```
(Q2')
select ?intersect where db:stop(?sid, ?intersect),
                        wfs:stop(?sid, ?geom),
                        filter(DWithin(?geom, (-88,43), 0.1)).
```

In this case, it is more efficient to query WFS feature stop for the stop ID where the geometry of the stop is with distance 0.1 of the point (-88,43). The retrieved stop ID is used to query database for the address of the stop intersection.

Lastly, Query Q3 can be rewritten as

```
(Q3') select ?g where wfs:stop(?id, ?geom),
                      filter(DWithin(?geom, (-88,43), 0.1)),
                      wfs:stop(?sid, ?g),
                      filter(DWithin(?g, ?geom, 1)).
```

In this case, two WFS requests are needed. The first request finds the geometry $?geom$ of a stop around the point (-88,43) and the second request finds all the stops that are within the distance of 1 from $?geom$. The geometries of all requested stops are returned.

Note that the execution of WFS and database queries may be more efficient if at least one variable of each WFS query is given values by the previous queries or such variable is constrained by some OGC filters. This is somewhat related to query rewriting in the context of data integration where data sources may have restrictions on possible access path to data [20].

Also note that in general, query rewriting using views should consider whether the rewritings are equivalent or maximally contained and whether the views are assumed to be complete or not [20]. In our case, the views (database or WFS schemas) on RDF schemas are complete and we seek equivalent rewriting. The views are relatively simple since there can be at most one database view and one WFS view on each RDF class. Because of these restrictions, each database table or WFS feature type has unique mapping in a RDF class. So if a RDF ontology R is created from a database and a WFS based on the above mapping and inference rules, and Q' is a rewriting of Q using Algorithm 2, then Q' should always produce the same result as executing Q on R. Also, though the complexity for query rewriting when queries and views are expressed as conjunctive queries is NP-complete [25], our algorithm is polynomial time because of the restrictions on views.

6.3 Semantic Constraints

RDF schema[10] includes some constructs of semantic constraints to assist ontology reasoning. The constructs include `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain`, and `rdfs:range`, etc. These restrictions can be useful in stating queries. For example, a RDF query for the instances of `SpatialFeature` class that are within a certain bounding box can be automatically translated to queries for instances of `Stop` and `Link` since they are both subclasses of `SpatialFeature`.

[10] http://www.w3.org/TR/rdf-schema/

In fact, the ontology defined in Figure 1 and 2 already contains some semantic constraints. The class `SpatialFeature` is a subclass of `Feature`, while the former has the subclasses `Link` and `Stop`, and the latter has the subclasses `Route`, `Pattern`, `LinkSequence`, etc. Also, we specified the domain and range of the property `route_link` to be `Route` and `Link`.

Because of these constraints some queries can be simplified. For example, Query `Q1` does not need to say that the type of `?route` is `Route` because the domain of `route_link` already has that restriction.

```
simplified (Q1)

select ?geom where ?route route_link ?link.
                   ?link geometry ?geom.
                   ?route name ?name.
                   filter(?name = "Summit").
```

Nothing needs to be changed in the algorithm for this example because the inference rule for `route_link` will add triples to specify the types of `?route` and `?link`.

In general, however, we need to include more inference rules to incorporate the use of semantic constraints. That is, we need to include inference rules to consider subclass and subproperty relations and the domain and range restrictions. We need to modify Algorithm 1 so that for each triple of the form `?s rdfs:type SpatialFeature`, we add additional triples `?s rdfs:type Stop` and `?s rdfs:type Link` since `Stop` and `Link` are subclasses of `SpatialFeature`. Similar treatment should be applied to `Feature`. Also, for any triple `?s p ?o`, if the property `p` has domain and range constraints, we should add type restrictions for `?s` and `?o`; if `p` has a subproperty `p'`, then we should add another triple `?s p' ?o`. Algorithm 2 remains the same.

More semantic constraints can be found in another ontology language OWL[11], an extension of RDF. For example, classes and properties in OWL can be declared as synonyms using `owl:sameAs`. Note that OWL-DL – a subset of OWL is based on description logics. Query rewriting with views is more difficult when the views are expressed in description logics and conjunctive queries over description logics. Query rewriting may be possible with some restrictions on the views [4]. Also, when description logics are combined with inference rules, the problem may be undecidable. Some restricted systems have decidable algorithms but they do not deal with query rewriting. Examples include $\mathcal{AL}$-Log [14], which is an integrated system for knowledge representation based on description logics and Datalog, and Motik *et.al.* [26] proposed a decidable combination of OWL-DL with function-free Horn rules.

7 Implementation Issues

Though the query rewriting algorithm can translate SPARQL queries to WFS requests and SQL queries, it is easier to do this using an existing application – D2R Server [7] that can create virtual RDF graphs from one or more relational databases. With D2R

[11] http://www.w3.org/TR/owl-features/

server, we no longer need to send SQL queries to databases directly. In fact, we can modify the Algorithm 2 slightly to make it work with D2R Server. D2R Server accepts SPARQL queries and through a set of mapping rules similar to what we have described, it translates the SPARQL queries to database queries and gets results back as RDF triples. So we first rewrite some RDF triples in the body of initial query to WFS getFeature requests. For example, Query Q1 can be rewritten to

```
     select ?geom where ?route rdf:type Route.
                        ?route route_link ?link.
                        ?route name ?name.
                        filter(?name = "Summit").
                        ?link  hasID ?lid.
                        wfs:link(?lid, ?geom).
```

Since we are interested in the link IDs of the route "Summit", we can send the following SPARQL query to the D2R Server:

```
(Q4) select ?lid where ?route rdf:type Route.
                       ?route route_link ?link.
                       ?route name ?name.
                       filter(?name = "Summit").
                       ?link  hasID ?lid.
```

The above query retrieves link IDs from database and we assume that the D2R Server has the same RDF ontology, which includes the RDF property `route_link`.

After obtaining the link IDs, a getFeature request similar to Figure 11 can be sent to WFS server to retrieve the geometries of the links. Notice that Query Q4 has an additional triple in its body `?link hasID ?lid` to retrieve link IDs. This can be generalized because the WFS features can be identified by their IDs. So in general, if the initial query involves a WFS feature, we require both the WFS getFeature request and the SPARQL query to D2R server to return IDs of the feature. If a WFS request is

```
<wfs:GetFeature service="WFS" version="1.0.0"
  xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:ogc="http://www.opengis.net/ogc" ...
  <wfs:Query typeName="wfs:link">
    <ogc:Filter>
       <OR>
          <PropertyIsEqualTo>
               <PropertyName>id</PropertyName>
               <Literal>22</Literal>
          </PropertyIsEqualTo>
          ...
       </OR>
    </ogc:Filter>
  </wfs:Query>
</wfs:GetFeature>
```

Fig. 11. A WFS GetFeature request to retrieve instances of feature Link with ID filters

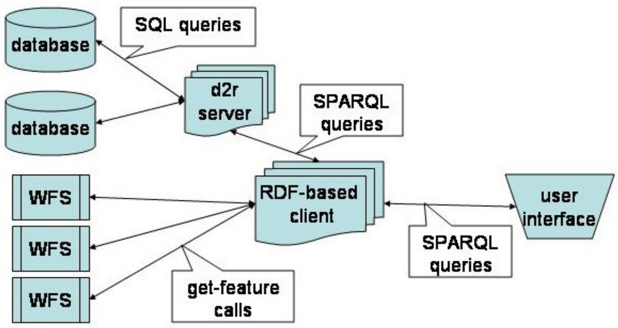


Fig. 12. RDF interface architecture

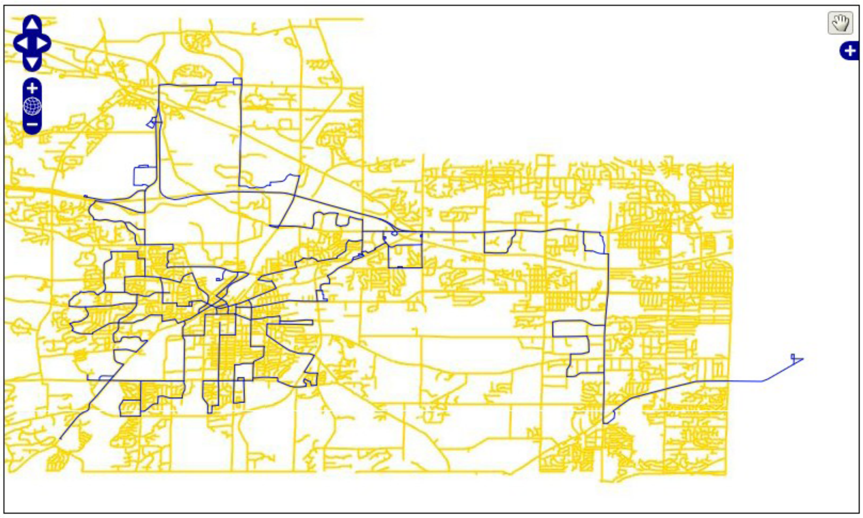
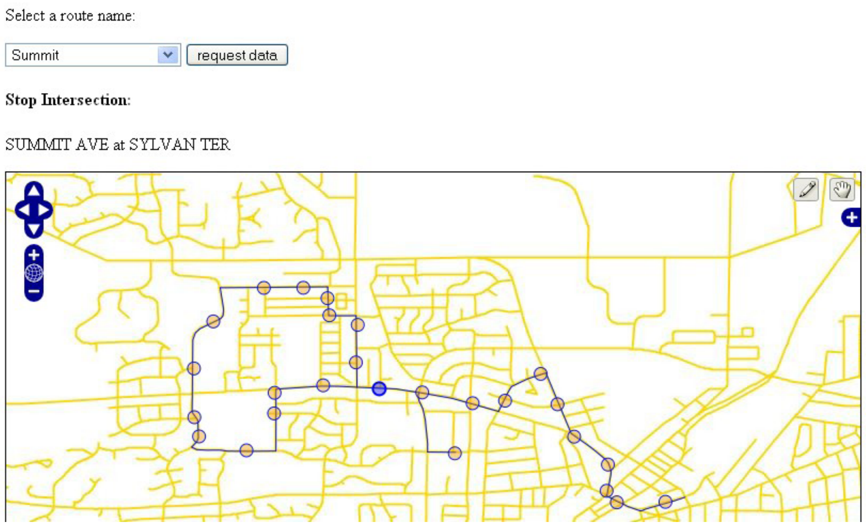


Fig. 13. Transit routes of Waukesha county

sent first, then the resulting feature IDs are used in the filters of the SPARQL query to D2R server. The architecture of the RDF interface is shown in Figure 12.

A sample demonstration of the interface[12] is shown in Figure 13, which shows Waukesha county's route links and streets as a WFS and a WMS layer, and Figure 14, which shows the links and bus stops of the route "Summit" retrieved from a WFS server using the link IDs and stop IDs retrieved from a D2R server. Also, as shown in Figure 14, users can click on a bus geometry (highlighted) and retrieve the intersection "SUMMIT AVE at SYLVAN TER" associated with the stop.

[12] http://jiangxi.cs.uwm.edu:8080/waukesha/gis.html


Select a route name:

Stop Intersection:

SUMMIT AVE at SYLVAN TER

Fig. 14. Route “Summit” and its stops

8 Conclusion and Future Work

We have presented a new method to provide integrated access to distributed geospatial data using RDF ontology and query rewriting. This method is more efficient than converting all data to ontology instances because it avoids the costs and consistency problems of data replication. Also, ontology interface can still use existing tools for conducting spatial query on geometries and relational queries on non-spatial data, and for rendering spatial features encoded in GML. Using the proposed method, user queries can be more straightforward because the application ontology can encode data semantics not available in WFS feature types or database schemas.

Our method uses RDF ontology but since RDF is a subset of OWL, the method can be directly applied to OWL ontology though extension may be needed to take advantage of the semantic constraints of OWL. OWL has more semantic constructs such as class equivalence/intersection/union, transitive/reflexive object properties, and the restrictions of universal/existential quantification on object properties. Our query rewriting algorithm needs to be extended to handle semantic constraints encoded with these constructs. However, it may be possible to use some existing tools such as Jena in this extension.

To evaluate the effectiveness of our method, we may need to implement a more complete ontology interface where user can query any classes and properties defined in domain and application ontology. The evaluation criteria may include the expressiveness of the interface – how many kinds of queries the interface can handle. The criteria should also include the efficiency of the interface – how fast the results are returned in comparison to the alternative ways of data integration such as data warehousing approach where all data are converted to ontology instances. Finally, we may want to evaluate the flexibility of the method – how easy it is to combine this method with other

GIS applications. For example, we can provide this ontology interface as a web service where other web-based applications can use it as a source of querying and rendering geospatial data.

As future work, we plan to implement the query rewriting algorithms to accept arbitrary SPARQL queries. Also, it may be useful to have ontology browser to display instances of ontology classes, thus users can navigate to other instances following links of object properties. The browser could be implemented on top of the query rewriting module. However, it is not clear how the contents of the browser should be presented since plain text of GML geometries is clearly not interesting while rendering all returned spatial instances could be very inefficient.

Acknowledgments

This research was supported in part by NSF award BCS-0616957. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors would also like to thank three anonymous reviewers for their valuable comments and suggestions

References

1. Budak Arpinar, I., Sheth, A., Ramakrishnan, C., Usery, E.L., Azami, M., Kwan, M.-P.: Geospatial ontology development and semantic analytics. Transactions in GIS 10(4), 551–575 (2006)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation and Application. Cambridge University Press, New York (2003)
3. Baglioni, M., Masserotti, M.V., Renso, C., Spinsanti, L.: Building geospatial ontologies from geographical databases. In: Proceedings of GeoSpatial Semantics, the Second International Conference, pp. 195–209 (2007)
4. Beeri, C., Levy, A.Y., Rousset, M.-C.: Rewriting queries using views in description logics. In: PODS 1997: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, pp. 99–108 (1997)
5. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American, 34–43 (2001)
6. Bishr, Y.: Overcoming the semantic and other barriers to GIS interoperability. International Journal of Geographical Information Science 12(4), 299–314 (1998)
7. Bizer, C., Seaborne, A.: D2RQ -treating Non-RDF databases as virtual RDF graphs. In: Proceedings of the 3rd International Semantic Web Conference (2004)
8. Chen, H., Wu, Z., Wang, H., Mao, Y.: RDF/RDFS-based relational database integration. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006), p. 94 (2006)
9. Choichi, J.: Constaint-based interoperability of spatiotemporal databases. Geoinformatica 3(3), 211–243 (1999)
10. Cruz, I.F., Sunna, W., Chaudhry, A.: Semi-automatic ontology alignment for geospatial data integration. In: Egenhofer, M.J., Freksa, C., Miller, H.J. (eds.) GIScience 2004. LNCS, vol. 3234, pp. 51–66. Springer, Heidelberg (2004)

11. OGC document 02-023r4. Geography Markup Language (GML), version 3.00 (2003)
12. OGC document 04-024. Web Map Service, Version 1.3 (2004)
13. OGC document 04-094. Web Feature Service, Version 1.1.0 (2005)
14. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A.: AL-log: Integrating datalog and description logics. Journal of Intelligent Information Systems 10(3), 227–252 (1998)
15. Egenhofer, M.: Toward the Semantic Geospatial Web. In: Tenth ACM International Symposium on Advances in Geographic Information Systems, pp. 1–4 (2002)
16. Fabrikant, S.I., Buttenfield, B.P.: Formalizing semantic spaces for information access. Annals of the Association of American Geographers 91 (2001)
17. FGDC. National Spatial Data Infrastructure (2007)
18. Fonseca, F., Egenhofer, M., Agouris, P., Camara, G.: Using ontologies for integrated geographic information systems. Transactions in GIS 6(3), 231–257 (2002)
19. Hobona, G., Fairbairn, G.D., James, P.: Semantically-assisted geospatial workflow design. In: Proceedings of 15th ACM International Symposium on Advances in Geographic Information Systems, November 2007, pp. 194–201 (2007)
20. Halevy, A.Y.: Theory of answering queries using views. SIGMOD Record (ACM Special Interest Group on Management of Data) 29(4), 40–47 (2000)
21. Halevy, A.Y.: Answering queries using views: A survey. VLDB Journal: Very Large Data Bases 10(4), 270–294 (2001)
22. Jones, C.B., Abdelmoty, A.I., Finch, D., Fu, G., Vaid, S.: The SPIRIT spatial search engine: architecture, ontologies and spatial indexing. In: Egenhofer, M.J., Freksa, C., Miller, H.J. (eds.) GIScience 2004. LNCS, vol. 3234, pp. 25–139. Springer, Heidelberg (2004)
23. Klien, E.: A rule-Based strategy for the semantic annotation of geodata. Transactions in GIS 11(3), 437–452 (2007)
24. Lenzerini, M.: Data integration: a theoretical perspective. In: PODS 2002: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 233–246 (2002)
25. Levy, A.Y., Mendelzon, A.O., Sagiv, Y., Srivastava, D.: Answering queries using views. In: Proceedings of the 14th ACM Symposium on Principles of Database Systems, pp. 95–104 (1995)
26. Motik, B., Sattler, U., Studer, R.: Query answering for owl-dl with rules. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 3(1), 41–60 (2005)
27. Paul, M., Ghosh, S.K.: An approach for service oriented discovery and retrieval of spatial data. In: Proceedings of the 2006 international workshop on Service-oriented software engineering, pp. 88–94 (2006)
28. Peng, Z.-R.: A proposed framework for featurelevel geospatial data sharing: A case study for transportation network data. International Journal of Geographic Information Sciences 19(4), 459–481 (2005)
29. Peng, Z.-R., Zhang, C.: The roles of geography markup language, scalable vector graphics, and web feature service specifications in the development of internet geographic information systems. Journal of Geographical Systems 6(2), 95–116 (2004)
30. Probst, F.: Ontological analysis of observations and measurements. In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 304–320. Springer, Heidelberg (2006)
31. Pundt, H., Bishr, Y.: Domain ontologies for data sharing - an example from environmental monitoring using field GIS. Computers and Geosciences 28(1), 95–102 (2002)
32. Smith, B., Mark, D.: Geographic categories: An ontological investigation. International Journal of Geographic Information Science 15(7), 591–612 (1998)

33. Sterling, L., Shapiro, E.: The Art of Prolog. MIT Press, Cambridge (1994)
34. Zhang, C., Li, W.: The roles of Web Feature and Web Map Services in real time geospatial data sharing for time-critical applications. Cartography and Geographic Information Science 32(4), 269–283 (2005)
35. Zhang, C., Li, W., Peng, Z.-R., Day, M.: GML-based interoperable geographical database. Cartography 32(2), 1–16 (2003)
36. Zhang, C., Li, W., Zhao, T.: Geospatial data sharing based on geospatial semantic web technologies. Journal of Spatial Science 52(2), 35–49 (2007)

Author Index